Discrete Optimization in a Nutshell

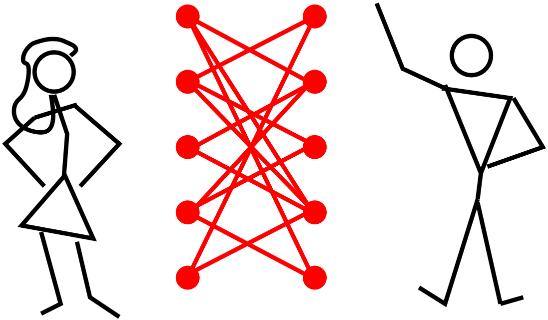
Christoph Helmberg

Integer Optimization

Contents

Contents

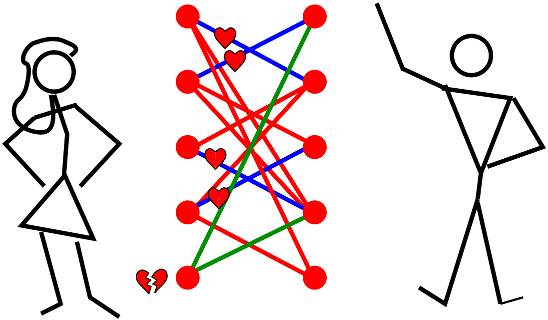
1.1 Application: The Marriage Problem (Bipartite Matching)



Find a maximum number of pairs!

men ↔ women, worker ↔ mashines, students ↔ positions, . . . (also weighted versions)

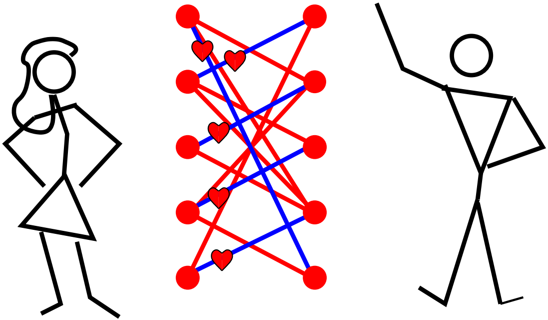
1.1 Application: The Marriage Problem (Bipartite Matching)



Find a maximum number of pairs!
Maximal (cannot be increased), but not of maximum cardinality

men ↔ women, worker ↔ mashines, students ↔ positions, . . . (also weighted versions)

1.1 Application: The Marriage Problem (Bipartite Matching)

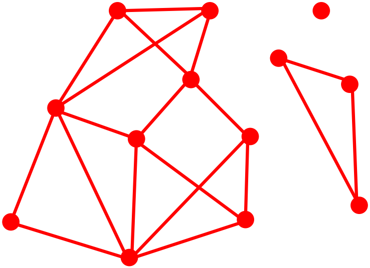


Find a maximum number of pairs!
Maximum Cardinality Matching (even perfect)

men ↔ women, worker ↔ mashines, students ↔ positions, . . . (also weighted versions)

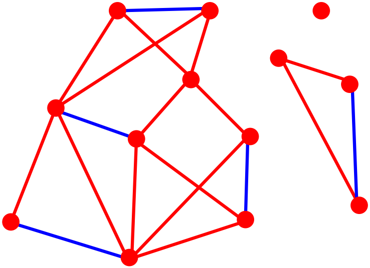
Bipartite Matching

- An (undirected) graph $G = (V, E)$ is a pair consisting of a node/vertex set V and an edge set $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Two nodes $u, v \in V$ are adjacent/neighbors, if $\{u, v\} \in E$.
- A node $v \in V$ and an edge $e \in E$ are incident, if $v \in e$.
- Two edges $e, f \in V$ are incident, if $e \cap f \neq \emptyset$.



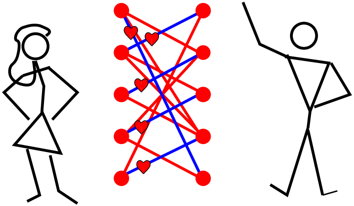
Bipartite Matching

- An (undirected) graph $G = (V, E)$ is a pair consisting of a node/vertex set V and an edge set $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Two nodes $u, v \in V$ are adjacent/neighbors, if $\{u, v\} \in E$.
- A node $v \in V$ and an edge $e \in E$ are incident, if $v \in e$.
- Two edges $e, f \in V$ are incident, if $e \cap f \neq \emptyset$.
- An edge set $M \subseteq E$ is a matching/1-factor, if for $e, f \in M$ with $e \neq f$ there holds $e \cap f = \emptyset$. The matching is perfect, if $|V| = 2|M|$.



Bipartite Matching

- An (undirected) graph $G = (V, E)$ is a pair consisting of a node/vertex set V and an edge set $E \subseteq \{\{u, v\} : u, v \in V, u \neq v\}$.
- Two nodes $u, v \in V$ are adjacent/neighbors, if $\{u, v\} \in E$.
- A node $v \in V$ and an edge $e \in E$ are incident, if $v \in e$.
- Two edges $e, f \in V$ are incident, if $e \cap f \neq \emptyset$.
- An edge set $M \subseteq E$ is a matching/1-factor, if for $e, f \in M$ with $e \neq f$ there holds $e \cap f = \emptyset$. The matching is perfect, if $|V| = 2|M|$.
- $G = (V, E)$ is bipartite, if $V = V_1 \cup V_2$ with $V_1 \cap V_2 = \emptyset$ and $E \subseteq \{\{u, v\} : u \in V_1, v \in V_2\}$.



Model: Maximum Cardinatliy Bipartite Matching

given: $G = (V_1 \dot{\cup} V_2, E)$ bipartite
find: matching $M \subseteq E$ with $|M|$ maximal

Model: Maximum Cardinatliy Bipartite Matching

given: $G = (V_1 \dot\cup V_2, E)$ bipartite
find: matching $M \subseteq E$ with $|M|$ maximal

---

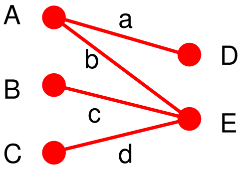
variables: $x \in \{0,1\}^E$ with $x_e = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{otherwise.} \end{cases} \quad (e \in E)$

(it represents the incidence/characteristic vector of M w.r.t. E)

constraints: $Ax \leq \mathbf{1}$,

where $A \in \{0,1\}^{V \times E}$ Node-Edge-Incidencematrix of G:

$$A_{v,e} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise.} \end{cases} \quad (v \in V, e \in E)$$



$$A = \begin{array}{c} \\ (A) \\ (B) \\ (C) \\ (D) \\ (E) \end{array} \left[\begin{array}{cccc} (a) & (b) & (c) & (d) \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \hline 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \end{array} \right]$$

Model: Maximum Cardinatliy Bipartite Matching

given: $G = (V_1 \dot\cup V_2, E)$ bipartite
find: matching $M \subseteq E$ with $|M|$ maximal

---

variables: $x \in \{0,1\}^E$ with $x_e = \begin{cases} 1 & \text{if } e \in M \\ 0 & \text{otherwise.} \end{cases} \quad (e \in E)$

(it represents the incidence/characteristic vector of M w.r.t. E)

constraints: $Ax \leq \mathbf{1}$,

where $A \in \{0,1\}^{V \times E}$ Node-Edge-Incidencematrix of G:

$$A_{v,e} = \begin{cases} 1 & \text{if } v \in e \\ 0 & \text{otherwise.} \end{cases} \quad (v \in V, e \in E)$$

---

optimization problem:
$$\begin{array}{ll} \max & \mathbf{1}^T x \\ \text{s.t.} & Ax \leq \mathbf{1} \\ & x \in \{0,1\}^E \end{array}$$

This is no LP! Enlarge $x \in \{0,1\}^E$ to $x \in [0,1]^E \to$ LP

For G bipartite, Simplex always delivers an optimal solution $x^* \in \{0,1\}^E$!
(this does not hold i.g. for general graphs G!)

Contents

1.2 Integeral Polyhedra

Simplex automatically yields an integral solution, if all vertices of the feasible set are integral.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Are all vertices of $\mathcal{X}$ integral?

1.2 Integeral Polyhedra

Simplex automatically yields an integral solution, if all vertices of the feasible set are integral.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Are all vertices of $\mathcal{X}$ integral? Almost never!
But there is an important class of matrices $A \in \mathbb{Z}^{m \times n}$, for which $\mathcal{X}$ has only integral vertices for any (!) $b \in \mathbb{Z}^m$:

A vertex is integral $\Leftrightarrow$ basic solution $x_B = A_B^{-1} b \in \mathbb{Z}^m$.
If $|\det(A_B)| = 1$, Cramer's rule implies $x_B \in \mathbb{Z}^m$.

1.2 Integeral Polyhedra

Simplex automatically yields an integral solution, if all vertices of the feasible set are integral.

$$\min c^T x \quad \text{s.t.} \quad x \in \mathcal{X} := \{x \geq 0 : Ax = b\}$$

Are all vertices of $\mathcal{X}$ integral? Almost never!
But there is an important class of matrices $A \in \mathbb{Z}^{m \times n}$, for which $\mathcal{X}$ has only integral vertices for any (!) $b \in \mathbb{Z}^m$:

A vertex is integral $\Leftrightarrow$ basic solution $x_B = A_B^{-1} b \in \mathbb{Z}^m$.
If $|\det(A_B)| = 1$, Cramer's rule implies $x_B \in \mathbb{Z}^m$.
A matrix $A \in \mathbb{Z}^{m \times n}$ of full row rank is unimodular, if $|\det(A_B)| = 1$ holds for each basis B.

Theorem

$A \in \mathbb{Z}^{m \times n}$ is unimodular if and only if for each $b \in \mathbb{Z}^m$ all vertices of the polyhedron $\mathcal{X} := \{x \geq 0 : Ax = b\}$ are integral.

Does this also hold for $\mathcal{X} := \{x \geq 0 : Ax \geq b\}$?

Totally Unimodular Matrices

$$\{x \geq 0 : Ax \geq b\} \quad \rightarrow \quad \left\{ \begin{bmatrix} x \\ s \end{bmatrix} \geq 0 : [A, I] \begin{bmatrix} x \\ s \end{bmatrix} = b \right\}$$

Certainly integral, if $\bar{A} = [A, I]$ is unimodular.
Laplace development of the determinant for each basis $B \rightarrow$

A matrix A is totally unimodular if for each square submatrix of A the determinant has value 0, 1 or -1. (requires $A \in \{0, 1, -1\}^{m \times n}$)

Theorem (Hoffmann und Kruskal)

$A \in \mathbb{Z}^{m \times n}$ is totally unimodular if and only if for each $b \in \mathbb{Z}^m$ all vertices of the polyhedron $\mathcal{X} := \{x \geq 0 : Ax \geq b\}$ are integral.

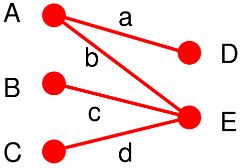
Note: A tot. unimod. $\Leftrightarrow$ A^T resp. $[A, -A, I, -I]$ tot. unimod.
consequence: dual LP, variants with equality constraints, etc. are integral

Recognizing Totally Unimodular Matrices

Theorem (Heller and Tompkins)

Let $A \in \{0, 1, -1\}^{m\times n}$ have at most two nonzero entries per column.
A is totally unimodular $\Leftrightarrow$ the rows A can be partitioned int two classes so that
(i) rows with one +1 and one -1 entry in the same column belong to the same class,
(ii) rows with two nonzeros of equal sign in the same column belong to distinct classes.

Example 1: the node-edge-incidencematrix of a bipartite graph



$$A = \left[\begin{array}{c|cccc} & (a) & (b) & (c) & (d) \\ (A) & 1 & 1 & 0 & 0 \\ (B) & 0 & 0 & 1 & 0 \\ (C) & 0 & 0 & 0 & 1 \\ \hline (D) & 1 & 0 & 0 & 0 \\ (E) & 0 & 1 & 1 & 1 \end{array}\right]$$

Example 1: Bipartite Graphs

A ... node-edge incidence matrix of $G = (V_1 \dot\cup V_2, E)$ bipartite

---

Bipartite Matching of Maximum Cardinality:

$$\max \mathbf{1}^T x \quad \text{s.t.} \quad Ax \le \mathbf{1},\ x \ge 0$$

Because A tot. unimod. the veritces of the feasilbe set are all integral
$\Rightarrow$ Simplex delivers optimal solution $x^* \in \{0,1\}^E$.

Example 1: Bipartite Graphs

A ... node-edge incidence matrix of $G = (V_1 \dot\cup V_2, E)$ bipartite

---

Bipartite Matching of Maximum Cardinality:

$$\max \mathbf{1}^T x \quad \text{s.t.} \quad Ax \leq \mathbf{1},\ x \geq 0$$

Because A tot. unimod. the veritces of the feasilbe set are all integral
$\Rightarrow$ Simplex delivers optimal solution $x^* \in \{0,1\}^E$.

The dual is also integral, because A^T tot. unimod.:

$$\min \mathbf{1}^T y \quad \text{s.t.} \quad A^T y \geq \mathbf{1},\ y \geq 0$$

Interpretation: $y^* \in \{0,1\}^V$ is the incidence vector of a smallest node set $V' \subseteq V$, so that $\forall e \in E : e \cap V' \neq \emptyset$ (**Minimum Vertex Cover**)

Example 1: Bipartite Graphs

A ... node-edge incidence matrix of $G = (V_1 \dot\cup V_2, E)$ bipartite

---

Bipartite Matching of Maximum Cardinality:

$$\max \mathbf{1}^T x \quad \text{s.t.} \quad Ax \leq \mathbf{1},\ x \geq 0$$

Because A tot. unimod. the veritces of the feasilbe set are all integral
$\Rightarrow$ Simplex delivers optimal solution $x^* \in \{0,1\}^E$.

The dual is also integral, because A^T tot. unimod.:

$$\min \mathbf{1}^T y \quad \text{s.t.} \quad A^T y \geq \mathbf{1},\ y \geq 0$$

Interpretation: $y^* \in \{0,1\}^V$ is the incidence vector of a smallest node set $V' \subseteq V$, so that $\forall e \in E : e \cap V' \neq \emptyset$ (**Minimum Vertex Cover**)

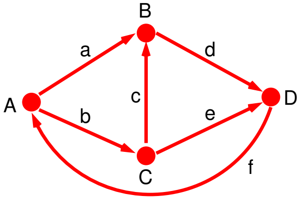
The **Assignment Problem**: $|V_1| = |V_2| = n$, complete bipartite:
$E = \{\{u,v\} : u \in V_1, v \in V_2\}$; edge weights $c \in \mathbb{R}^E$
Find a perfect matching of minimum total weight:

$$\min c^T x \quad \text{s.t.} \quad Ax = \mathbf{1},\ x \geq 0$$

is integral, too, because $[A; -A]$ tot. unimod.

Example 2: Node-Arc Incidence Matrix of Digraphs

- A digraph/directed graph $D = (V, E)$ is a pair consisting of a node set V and a (multi-)set of directed edges/arcs $E \subseteq \{(u, v) : u, v \in V, u \neq v\}$. [multiple arcs are allowed!]
- For $e = (u, v) \in E$, u is the tail and v the head of e.

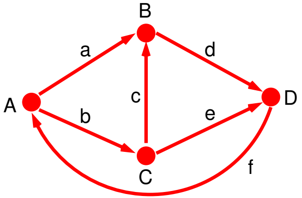


Example 2: Node-Arc Incidence Matrix of Digraphs

- A digraph/directed graph $D = (V, E)$ is a pair consisting of a node set V and a (multi-)set of directed edges/arcs $E \subseteq \{(u, v) : u, v \in V, u \neq v\}$. [multiple arcs are allowed!]
- For $e = (u, v) \in E$, u is the tail and v the head of e.
- The node-arc incidence matrix $A \in \{0, 1, -1\}^{V \times E}$ of D has entries

$$A_{v,e} = \begin{cases} -1 & v \text{ is the tail of } e \\ 1 & v \text{ is the head of } e \\ 0 & \text{otherwise} \end{cases} \qquad (v \in V, e \in E).$$

the node-arc incidence matrix of a digraph is totally unimodular.



$$A = \begin{bmatrix} & (a) & (b) & (c) & (d) & (e) & (f) \\ (A) & -1 & -1 & 0 & 0 & 0 & 1 \\ (B) & 1 & 0 & 1 & -1 & 0 & 0 \\ (C) & 0 & 1 & -1 & 0 & -1 & 0 \\ (D) & 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

Contents

1.3 Application: Networkflow

Modelling tool: transport problems, evacuation planning, scheduling, (internet) traffic planning, ...

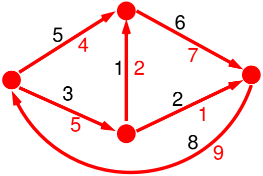
- A network (D, w) consists of a digraph $D = (V, E)$ and (arc-)capacities $w \in \mathbb{R}^E_+$.
- A vector $x \in \mathbb{R}^E$ is a flow on (D, w), if it satisfies the

flow conservation constraints $$\sum_{e=(u,v)\in E} x_e = \sum_{e=(v,u)\in E} x_e \quad (v \in V)$$

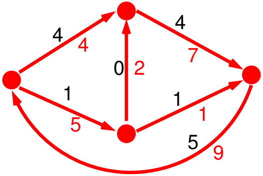
[$\Leftrightarrow$ $Ax = 0$ for node-arc incidence matrix A]

- A flow $x \in \mathbb{R}^E$ on (D,w) is feasible, if $0 \le x \le w$

[lower bounds would also be possible]



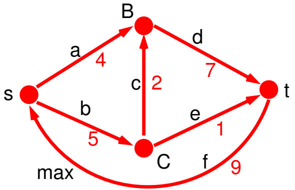
flow



feasible flow

Maximal s-t-Flows, Minimal s-t-Cuts

Given a source $s \in V$ and a sink $t \in V$ with $(t, s) \in E$, find a feasible flow $x \in \mathbb{R}^E$ on (D, w) with maximum flow value $x_{(t,s)}$.



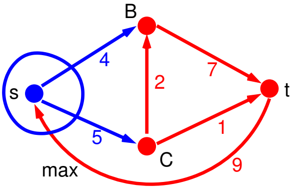
$$A = \begin{array}{ccccccc} & (a) & (b) & (c) & (d) & (e) & (f) \\ (s) & -1 & -1 & 0 & 0 & 0 & 1 \\ (B) & 1 & 0 & 1 & -1 & 0 & 0 \\ (C) & 0 & 1 & -1 & 0 & -1 & 0 \\ (t) & 0 & 0 & 0 & 1 & 1 & -1 \end{array}$$

LP: $\max \ x_{(t,s)}$ s.t. $Ax = 0, \ 0 \le x \le w,$

If $w \in \mathbb{Z}^E$, the OS of Simplex is $x^* \in \mathbb{Z}^E$, because $[A; -A; I]$ tot. unimod.

Maximal s-t-Flows, Minimal s-t-Cuts

Given a source $s \in V$ and a sink $t \in V$ with $(t, s) \in E$, find a feasible flow $x \in \mathbb{R}^E$ on (D, w) with maximum flow value $x_{(t,s)}$.



$S = \{s\}$,
$\delta^+(S) = \{(s, B), (s, C)\}$,
$w(\delta^+(S)) = 4 + 5 = 9$.

LP: $\max\ x_{(t,s)}$ s.t. $Ax = 0,\ 0 \le x \le w$,

If $w \in \mathbb{Z}^E$, the OS of Simplex is $x^* \in \mathbb{Z}^E$, because $[A; -A; I]$ tot. unimod.

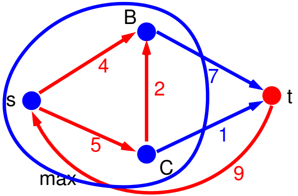
Each $S \subseteq V$ with $s \in S$ and $t \notin S$ defines an s-t-cut

$$\delta^+(S) := \{(u, v) \in E : u \in S, v \notin S\},$$

the out-flow is at most $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, the value of the cut.

Maximal s-t-Flows, Minimal s-t-Cuts

Given a source $s \in V$ and a sink $t \in V$ with $(t, s) \in E$, find a feasible flow $x \in \mathbb{R}^E$ on (D, w) with maximum flow value $x_{(t,s)}$.



$S = \{s, B, C\}$,
$\delta^+(S) = \{(B, t), (C, t)\}$,
$w(\delta^+(S)) = 7 + 1 = 8$.

LP: $\max\ x_{(t,s)}$ s.t. $Ax = 0,\ 0 \leq x \leq w$,

If $w \in \mathbb{Z}^E$, the OS of Simplex is $x^* \in \mathbb{Z}^E$, because $[A; -A; I]$ tot. unimod.

Each $S \subseteq V$ with $s \in S$ and $t \notin S$ defines an s-t-cut

$$\delta^+(S) := \{(u, v) \in E : u \in S, v \notin S\},$$

the out-flow is at most $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, the value of the cut.

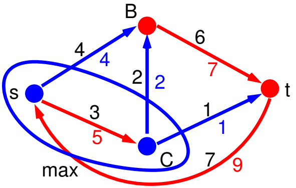
Theorem (Max-Flow Min-Cut Theorem of Ford and Fulkerson)

The maximum s-t-flow value equals the minimum s-t cut value.

[both computable by Simplex]

Maximal s-t-Flows, Minimal s-t-Cuts

Given a source $s \in V$ and a sink $t \in V$ with $(t,s) \in E$, find a feasible flow $x \in \mathbb{R}^E$ on (D,w) with maximum flow value $x_{(t,s)}$.



$S = \{s, C\}$,
$\delta^+(S) = \{(s,B),(C,B),(C,t)\}$,
$w(\delta^+(S)) = 4+2+1 = 7 = x^*_{(t,s)}$

LP: $\max\ x_{(t,s)}$ s.t. $Ax = 0,\ 0 \le x \le w$,

If $w \in \mathbb{Z}^E$, the OS of Simplex is $x^* \in \mathbb{Z}^E$, because $[A; -A; I]$ tot. unimod.

Each $S \subseteq V$ with $s \in S$ and $t \notin S$ defines an s-t-cut

$$\delta^+(S) := \{(u,v) \in E : u \in S, v \notin S\},$$

the out-flow is at most $w(\delta^+(S)) := \sum_{e \in \delta^+(S)} w_e$, the value of the cut.

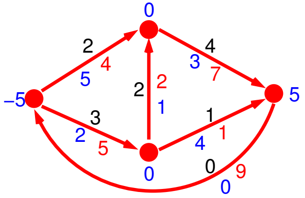
Theorem (Max-Flow Min-Cut Theorem of Ford and Fulkerson)

The maximum s-t-flow value equals the minimum s-t cut value.

[both computable by Simplex]

Minimum Cost Flow (Min-Cost-Flow)

The flow value is now prescribed by balances $b \in \mathbb{R}^V$ ($\mathbf{1}^T b = 0$) on the nodes; each unit of flow induces arc costs $c \in \mathbb{R}^E$.
Find the cheapest flow.



$$\begin{array}{ll} \min & \begin{bmatrix} 5 & 2 & 1 & 3 & 4 & 0 \end{bmatrix} x \\ \text{s.t.} & \begin{bmatrix} -1 & -1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} x = \begin{bmatrix} 5 \\ 0 \\ 0 \\ -5 \end{bmatrix} \\ & 0 \le x \le w \end{array}$$

LP: $\min\ c^T x \quad \text{s.t.} \quad Ax = b,\ 0 \le x \le w,$

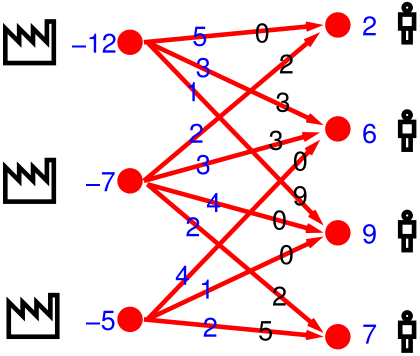
For b, c and w integr., Simplex gives OS $x^* \in \mathbb{Z}^E$, as $[A; -A; I]$ tot. unimod.
[also works for lower bounds on arcs: $u \le x \le w$!]

For LPs $\min\ c^T x \quad \text{s.t.} \quad Ax = b,\ u \le x \le w,$ A node-arc inc.
there is a particularly efficient simplex variant, the network simplex,
it only needs addtions, subtractions and comparisons!

Extremly broad scope of applications, popular modelling tool

Example: Transportation Problem

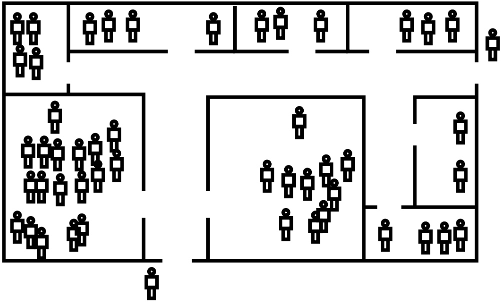
A company with several production sites has to serve several customers.
How is this best done in view of transportation costs?



Note: only one product!

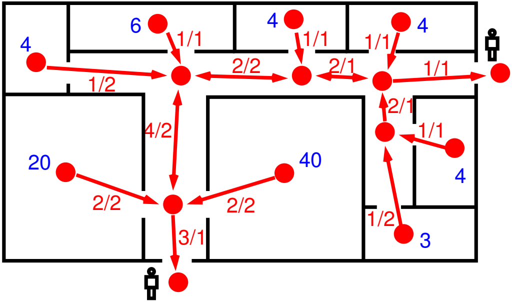
Example: Evacuation Planning

Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



Example: Evacuation Planning

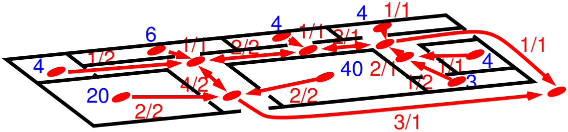
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



Arcs with capacities and crossing times

Example: Evacuation Planning

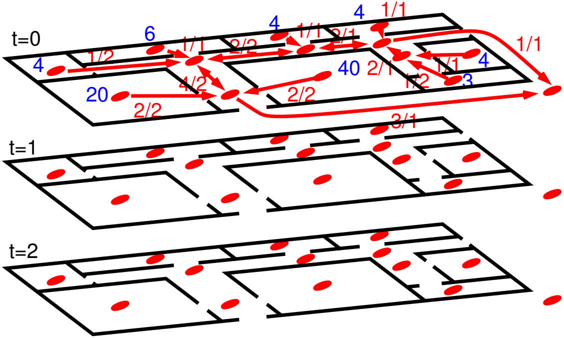
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



geometry is not important, simplify

Example: Evacuation Planning

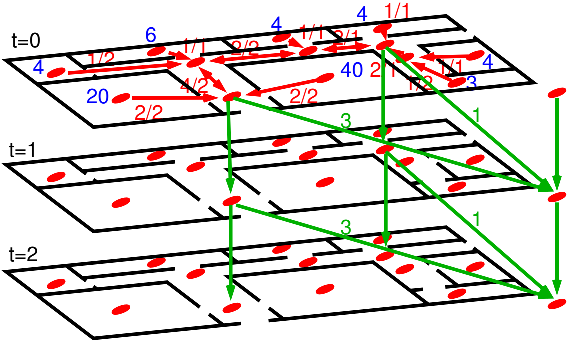
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



discretize time, one level per time step

Example: Evacuation Planning

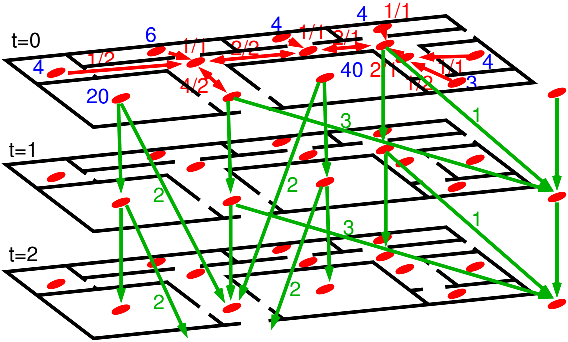
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



crossing times connect levels, capacity stays

Example: Evacuation Planning

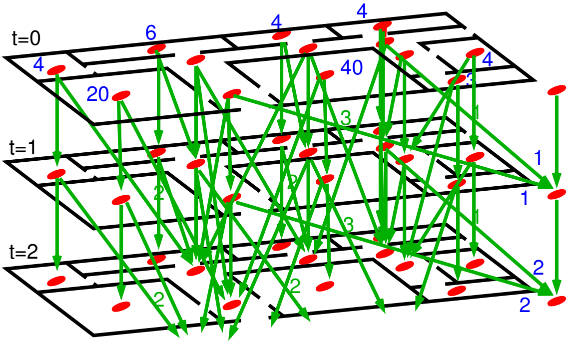
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



crossing times connects levels, capacity stays

Example: Evacuation Planning

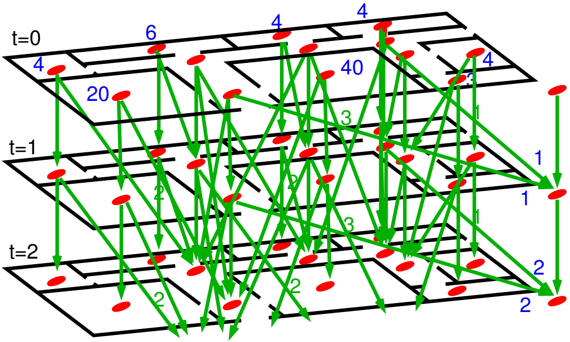
Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



rising costs on the exit arcs to encourage fast exits

Example: Evacuation Planning

Determine for each room a flight path, so that the building is cleared as fast as possible. Per aisle capacity and crossing times are known.



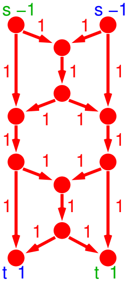
rising costs on the exit arcs to encourage fast exits

Approach only ok if persons do not need to be discerned!

Contents

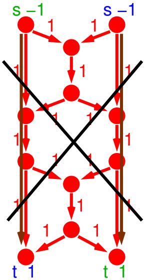
1.4 Multi-Commodity Flow Problems

Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.



1.4 Multi-Commodity Flow Problems

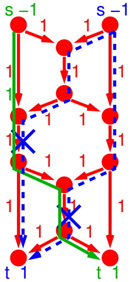
Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.



mixing is forbidden!

1.4 Multi-Commodity Flow Problems

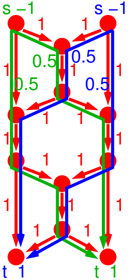
Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.



integr. is impossible

1.4 Multi-Commodity Flow Problems

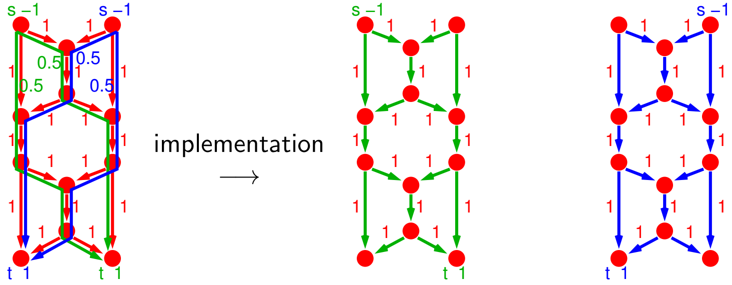
Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.



fractional works

1.4 Multi-Commodity Flow Problems

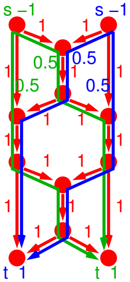
Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.



fractional works copy 1 copy 2

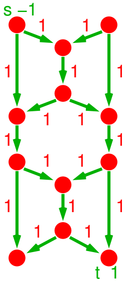
1.4 Multi-Commodity Flow Problems

Given a network (D, w) and several different commodities $K = \{1, \ldots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.

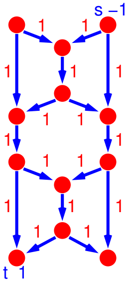


fractional works

implementation $\longrightarrow$



copy 1

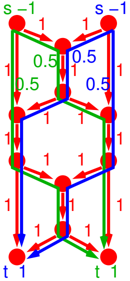


copy 2

$$\begin{array}{lcccccc}
\min & {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} & & \\
\text{s.t.} & A x^{(1)} & & & = & b^{(1)} \\
 & & & A x^{(2)} & = & b^{(2)} \\
 & I x^{(1)} & + & I x^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, & & x^{(2)} \geq 0. & &
\end{array}$$

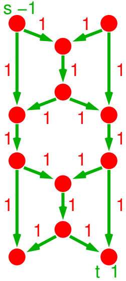
1.4 Multi-Commodity Flow Problems

Given a network (D, w) and several different commodities $K = \{1, \dots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.

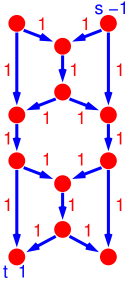


fractional works

implementation $\longrightarrow$



copy 1



copy 2

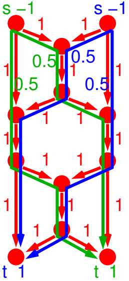
$$\begin{bmatrix} A & 0 \\ 0 & A \\ I & I \end{bmatrix}$$

i.g. not tot.unimod.!

$$\begin{array}{llcllcl}
\min & {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} & & & \\
\text{s.t.} & A x^{(1)} & & & & = & b^{(1)} \\
& & & A x^{(2)} & & = & b^{(2)} \\
& I x^{(1)} & + & I x^{(2)} & & \leq & w \\
& x^{(1)} \geq 0, & & x^{(2)} \geq 0. & & &
\end{array}$$

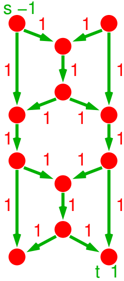
1.4 Multi-Commodity Flow Problems

Given a network (D, w) and several different commodities $K = \{1, \dots, k\}$ with sources/sinks (s_i, t_i) and flow values f_i, $i \in K$ find feasible flows $x^{(i)} \in \mathbb{R}^E$, so that in sum capacities are observed.

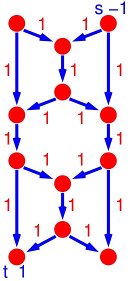


fractional works

implementation $\longrightarrow$



copy 1



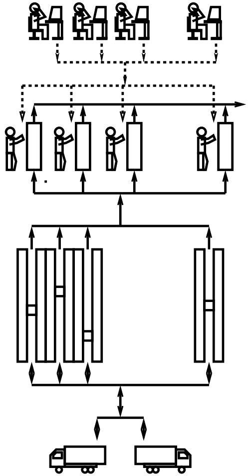
copy 2

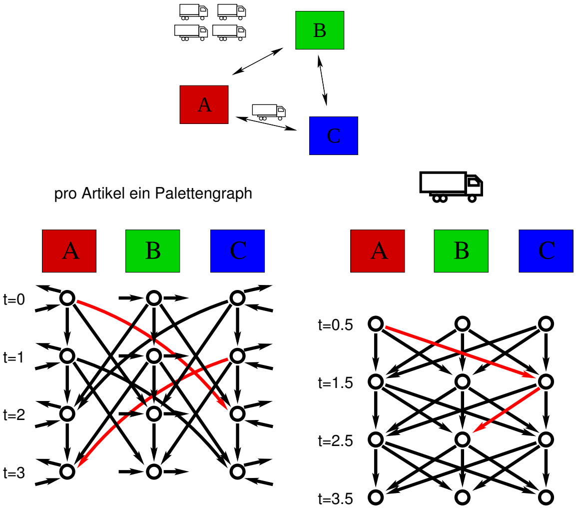
fractionally solvable,
integral VERY difficult!

$$
\begin{array}{lccccc}
\min & {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} & & \\
\text{s.t.} & A x^{(1)} & & & = & b^{(1)} \\
 & & & A x^{(2)} & = & b^{(2)} \\
 & I x^{(1)} & + & I x^{(2)} & \leq & w \\
 & x^{(1)} \geq 0, & & x^{(2)} \geq 0. & &
\end{array}
$$

Example: Logistics

cover demand by shifting pallets by trucks between warehouses





Further Application Areas

$\left.\begin{array}{l}\bullet\ \text{fractional: capacity planning}\\ \bullet\ \text{integral: time discretized routing and scheduling}\end{array}\right\}$ for

- fractional: capacity planning
- integral: time discretized routing and scheduling
  - street traffic
  - trains
  - internet
  - logistics (bottleneck analysis/steering)
  - production (mashine loads/-assignment)

Further Application Areas

$\left.\begin{array}{l}\text{• fractional: capacity planning} \\ \text{• integral: time discretized routing and scheduling}\end{array}\right\}$ for

- street traffic
- trains
- internet
- logistics (bottleneck analysis/steering)
- production (mashine loads/-assignment)

---

network-design:
installed capacities should satisfy as many demands as possible also in case of failures [“robust” variants are extremly difficult!]

Multi-commodity flow is often used as basic model that is combined with further constraints.

Contents

1.5 Integer Optimization (Integer Programming)

mainly: linear programs with exclusively integer variables
(otherw. mixed integer programming)

$$\begin{array}{ll} \max & c^T x \\ \text{s.t.} & Ax \le b \\ & x \in \mathbb{Z}^n \end{array}$$

Typically contains many binary variables ($\{0,1\}$) for yes/no decisions

---

Difficulty: i.g. not solvable “efficiently”, complexity class *NP*
$\Rightarrow$ exact solutions rely heavily on enumeration (systematic exploration)

Exact solutions by combining the following techniques:

- (upper) bounds by linear/convex relaxation
  improved by cutting plane approaches
- feasible solutions (lower bounds) by rounding- and
  local search heuristics
- enumerate by branch&bound, branch&cut

Combinatorial Optimization

Mathematically: Given a finite ground set Ω, a set of feasible subsets $\mathcal{F} \subseteq 2^{\Omega}$[power set, set of all subsets] and a goal function $c : \mathcal{F} \to \mathbb{Q}$, determine

$$\max\{c(F) : F \in \mathcal{F}\} \quad \text{or} \quad F \in \text{Argmax}\{c(F) : F \in \mathcal{F}\}$$

Combinatorial Optimization

Mathematically: Given a finite ground set Ω, a set of feasible subsets $\mathcal{F} \subseteq 2^{\Omega}$[power set, set of all subsets] and a goal function $c : \mathcal{F} \to \mathbb{Q}$, determine

$$\max\{c(F) : F \in \mathcal{F}\} \quad \text{or} \quad F \in \text{Argmax}\{c(F) : F \in \mathcal{F}\}$$

Mostly only linear c: $c(F) := \sum_{e \in F} c_e$ with $c \in \mathbb{Q}^{\Omega}$.

Combinatorial Optimization

Mathematically: Given a finite ground set Ω, a set of feasible subsets $\mathcal{F} \subseteq 2^{\Omega}$[power set, set of all subsets] and a goal function $c : \mathcal{F} \to \mathbb{Q}$, determine

$$\max\{c(F) : F \in \mathcal{F}\} \quad \text{or} \quad F \in \operatorname{Argmax}\{c(F) : F \in \mathcal{F}\}$$

Mostly only linear c: $c(F) := \sum_{e \in F} c_e$ with $c \in \mathbb{Q}^{\Omega}$.

Ex1: maximum cardinality matching for $G = (V, E)$:
$\Omega = E$, $\mathcal{F} = \{M \subseteq E : M \text{ matching in } G\}$, $c = \mathbf{1}$

Ex2: minimum vertex cover for $G = (V, E)$:
$\Omega = V$, $\mathcal{F} = \{V' \subseteq V : e \cap V' \neq \emptyset \text{ für } e \in E\}$, $c = -\mathbf{1}$

Combinatorial Optimization

Mathematically: Given a finite ground set Ω, a set of feasible subsets $\mathcal{F} \subseteq 2^{\Omega}$[power set, set of all subsets] and a goal function $c : \mathcal{F} \to \mathbb{Q}$, determine

$$\max\{c(F) : F \in \mathcal{F}\} \quad \text{or} \quad F \in \operatorname{Argmax}\{c(F) : F \in \mathcal{F}\}$$

Mostly only linear c: $c(F) := \sum_{e \in F} c_e$ with $c \in \mathbb{Q}^{\Omega}$.

Ex1: maximum cardinality matching for $G = (V, E)$:
$\Omega = E$, $\mathcal{F} = \{M \subseteq E : M \text{ matching in } G\}$, $c = \mathbf{1}$

Ex2: minimum vertex cover for $G = (V, E)$:
$\Omega = V$, $\mathcal{F} = \{V' \subseteq V : e \cap V' \neq \emptyset \text{ für } e \in E\}$, $c = -\mathbf{1}$

---

formulation as a binary program:
Notation: incidence-/characteristic vector $\chi_{\Omega}(F) \in \{0, 1\}^{\Omega}$ for $F \subseteq \Omega$
[in short $\chi(F)$, satisfies $[\chi(F)]_e = 1 \Leftrightarrow e \in F$]
A linear program $\max\{c^T x : Ax \leq b, x \in [0, 1]^{\Omega}\}$ is a formulation of the combinatorial optimization problem, if

$$\{x \in \{0, 1\}^{\Omega} : Ax \leq b\} = \{\chi(F) : F \in \mathcal{F}\}.$$

(Algorithmic) Complexity of Problems

An instance I of a problem is a concrete assignment of values to the problem data; its size $|I|$ is the encoding length, i.e. the number of symbols in a description string according to a reasonalbe encoding scheme.

(Algorithmic) Complexity of Problems

An instance I of a problem is a concrete assignment of values to the problem data; its size $|I|$ is the encoding length, i.e. the number of symbols in a description string according to a reasonalbe encoding scheme.

The runtime of an algorithm for one instance is the number of elementary operations executed (read/write a symbol, add/multiply/compare bytes, etc.)

(Algorithmic) Complexity of Problems

An instance I of a problem is a concrete assignment of values to the problem data; its size $|I|$ is the encoding length, i.e. the number of symbols in a description string according to a reasonalbe encoding scheme.

The runtime of an algorithm for one instance is the number of elementary operations executed (read/write a symbol, add/multiply/compare bytes, etc.)

An algorithm solves a problem in polynomial time or efficiently, if it computes the correct answer within a runtime that is bounded by a polynomial in the encoding length.

(Algorithmic) Complexity of Problems

An instance I of a problem is a concrete assignment of values to the problem data; its size $|I|$ is the encoding length, i.e. the number of symbols in a description string according to a reasonalbe encoding scheme.

The runtime of an algorithm for one instance is the number of elementary operations executed (read/write a symbol, add/multiply/compare bytes, etc.)

An algorithm solves a problem in polynomial time or efficiently, if it computes the correct answer within a runtime that is bounded by a polynomial in the encoding length.

A problem is polynomially/efficiently solvable, if there is an algorithm that solves it efficiently. The class P comprises all problems, that are solvable efficiently.

(Algorithmic) Complexity of Problems

An instance I of a problem is a concrete assignment of values to the problem data; its size $|I|$ is the encoding length, i.e. the number of symbols in a description string according to a reasonalbe encoding scheme.

The runtime of an algorithm for one instance is the number of elementary operations executed (read/write a symbol, add/multiply/compare bytes, etc.)

An algorithm solves a problem in polynomial time or efficiently, if it computes the correct answer within a runtime that is bounded by a polynomial in the encoding length.

A problem is polynomially/efficiently solvable, if there is an algorithm that solves it efficiently. The class P comprises all problems, that are solvable efficiently.

Ex1: lineare optimization is in P, BUT Simplex is not polynomial.
Ex2: maximum cardinality matching in general graphs is in P.
Ex3: minimum vertex cover in bipartite graphs is in P.

Decision Problems and the Class *NP*

In a decision problem each instance is a question, that must be answered by either "yes" or "no".

A decision problem is solvable in nondeterministic polynomial time, if for each "yes"-instance I a solution string (the certificate) exists, that allows to check correctnes of the "yes"-answer in time polynomially bounded in $|I|$. [only "yes" is relevant!]

Decision Problems and the Class NP

In a decision problem each instance is a question, that must be answered by either “yes” or “no”.

A decision problem is solvable in nondeterministic polynomial time, if for each “yes”-instance I a solution string (the certificate) exists, that allows to check correctnes of the “yes”-answer in time polynomially bounded in $|I|$. [only “yes” is relevant!]

The cass NP comprises all decision problems that are solvable in nondeterministic polynomial time. There holds: $P \subseteq NP$

Decision Problems and the Class *NP*

In a decision problem each instance is a question, that must be answered by either "yes" or "no".

A decision problem is solvable in nondeterministic polynomial time, if for each "yes"-instance I a solution string (the certificate) exists, that allows to check correctnes of the "yes"-answer in time polynomially bounded in $|I|$. [only "yes" is relevant!]

The cass NP comprises all decision problems that are solvable in nondeterministic polynomial time. There holds: $P \subseteq NP$

Ex: **Hamiltonian Cycle:** in a graphen $G = (V, E)$ an edge set $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ is a cycle (of length k) if $v_i \neq v_j$ for $i \neq j$.
A cycle C is hamiltonian if $|C| = |V|$.

Decision Problems and the Class *NP*

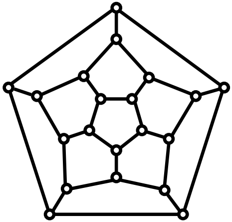
In a decision problem each instance is a question, that must be answered by either “yes” or “no”.

A decision problem is solvable in nondeterministic polynomial time, if for each “yes”-instance I a solution string (the certificate) exists, that allows to check correctnes of the “yes”-answer in time polynomially bounded in $|I|$. [only “yes” is relevant!]

The cass NP comprises all decision problems that are solvable in nondeterministic polynomial time. There holds: $P \subseteq NP$

Ex: **Hamiltonian Cycle:** in a graphen $G = (V, E)$ an edge set $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ is a cycle (of length k) if $v_i \neq v_j$ for $i \neq j$.
A cycle C is hamiltonian if $|C| = |V|$.

decision problem:
Does G contain a Hamiltonian cycle?



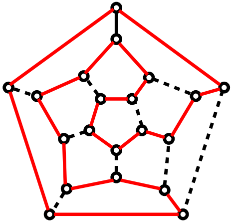
Decision Problems and the Class *NP*

In a decision problem each instance is a question, that must be answered by either "yes" or "no".

A decision problem is solvable in nondeterministic polynomial time, if for each "yes"-instance I a solution string (the certificate) exists, that allows to check correctnes of the "yes"-answer in time polynomially bounded in $|I|$. [only "yes" is relevant!]

The cass NP comprises all decision problems that are solvable in nondeterministic polynomial time. There holds: $P \subseteq NP$

Ex: **Hamiltonian Cycle:** in a graphen $G = (V, E)$ an edge set $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ is a cycle (of length k) if $v_i \neq v_j$ for $i \neq j$.
A cycle C is hamiltonian if $|C| = |V|$.



decision problem:
Does G contain a Hamiltonian cycle?

Yes-answer can be checked efficiently, the problem is in NP.

Decision Problems and the Class NP

In a decision problem each instance is a question, that must be answered by either "yes" or "no".

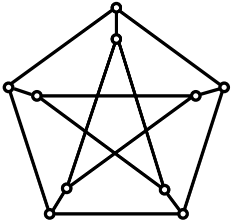
A decision problem is solvable in nondeterministic polynomial time, if for each "yes"-instance I a solution string (the certificate) exists, that allows to check correctnes of the "yes"-answer in time polynomially bounded in $|I|$. [only "yes" is relevant!]

The cass NP comprises all decision problems that are solvable in nondeterministic polynomial time. There holds: $P \subseteq NP$

Ex: **Hamiltonian Cycle:** in a graphen $G = (V, E)$ an edge set $C = \{\{v_1, v_2\}, \{v_2, v_3\}, \dots, \{v_{k-1}, v_k\}, \{v_k, v_1\}\} \subseteq E$ is a cycle (of length k) if $v_i \neq v_j$ for $i \neq j$.
A cycle C is hamiltonian if $|C| = |V|$.

decision problem:
Does G contain a Hamiltonian cycle?

Yes-answer can be checked efficiently, the problem is in NP.



NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

If P_1 can be polynomially transformed to P_2 then an efficient algorithm for P_2 also solves P_1 efficiently; P_2 is at least as difficult as P_1.

NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

If P_1 can be polynomially transformed to P_2 then an efficient algorithm for P_2 also solves P_1 efficiently; P_2 is at least as difficult as P_1.

If $\bar{P} \in NP$ and each $\hat{P} \in NP$ is polynomially transformable to $\bar{P}$ then $\bar{P}$ is NP-complete.

NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

If P_1 can be polynomially transformed to P_2 then an efficient algorithm for P_2 also solves P_1 efficiently; P_2 is at least as difficult as P_1.

If $\bar{P} \in NP$ and each $\hat{P} \in NP$ is polynomially transformable to $\bar{P}$ then $\bar{P}$ is *NP*-complete.

If an *NP*-complete problem $\bar{P}$ can be polynomially transformed to a problem $\hat{P} \in NP$ then $\hat{P}$ is *NP*-complete, too; so they are equally difficult.

NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

If P_1 can be polynomially transformed to P_2 then an efficient algorithm for P_2 also solves P_1 efficiently; P_2 is at least as difficult as P_1.

If $\bar{P} \in NP$ and each $\hat{P} \in NP$ is polynomially transformable to $\bar{P}$ then $\bar{P}$ is NP-complete.

If an NP-complete problem $\bar{P}$ can be polynomially transformed to a problem $\hat{P} \in NP$ then $\hat{P}$ is NP-complete, too; so they are equally difficult.

There are voluminous collections of NP-complete problems; examples:

- integer optimization (in its decision version)
- integer multi-commodity flow
- Hamiltonian Cycle
- Minimum Vertex Cover on general graphs
- Knapsack (for big numbers)

NP-complete Problems

A decision problem P_1 can be polynomially transformed to a decision problem P_2 if there is an algorithm that transforms each instance I_1 of P_1 in running time polynomial in $|I_1|$ into an instance I_2 of P_2 so that I_2 is a yes-instance of P_2 if and only if I_1 is a yes-instance of P_1.

If P_1 can be polynomially transformed to P_2 then an efficient algorithm for P_2 also solves P_1 efficiently; P_2 is at least as difficult as P_1.

If $\bar{P} \in NP$ and each $\hat{P} \in NP$ is polynomially transformable to $\bar{P}$ then $\bar{P}$ is NP-complete.

If an NP-complete problem $\bar{P}$ can be polynomially transformed to a problem $\hat{P} \in NP$ then $\hat{P}$ is NP-complete, too; so they are equally difficult.

If there is an efficient algorithm for one NP-complete problem, all are solvable efficiently. For years the assumption is: $P \neq NP$.

If one wants to solve all instances of a problem, partial enumeration seems to be unavoidable.

A problem is NP-hard, if it would allow to solve an NP-complete problem.

Contents

1.6 Branch-and-Bound

In enumerating all solutions, as many as possible should be eliminated early on by upper and lower bounds.

1.6 Branch-and-Bound

In enumerating all solutions, as many as possible should be eliminated early on by upper and lower bounds.

Ex: $\{0,1\}$-knapsack: weights $a \in \mathbb{N}^n$, capacity $b \in \mathbb{N}$, profit $c \in \mathbb{N}^n$,

$$\max\ c^T x \quad \text{s.t.} \quad a^T x \leq b,\ x \in \{0,1\}^n$$

upper bound: max $c^T x$ s.t. $a^T x \leq b,\ x \in [0,1]^n$ [LP-relaxation]
lower bound: sort by profit/weight and fill in this sequence

1.6 Branch-and-Bound

In enumerating all solutions, as many as possible should be eliminated early on by upper and lower bounds.

Ex: $\{0,1\}$-knapsack: weights $a \in \mathbb{N}^n$, capacity $b \in \mathbb{N}$, profit $c \in \mathbb{N}^n$,

$$\max\ c^T x \quad \text{s.t.} \quad a^T x \leq b,\ x \in \{0,1\}^n$$

upper bound: max $c^T x$ s.t. $a^T x \leq b,\ x \in [0,1]^n$ [LP-relaxation]
lower bound: sort by profit/weight and fill in this sequence

---

algorithmic scheme (for maximization problems):
M ... set of open problems, initially $M = \{\text{orig. problem}\}$
$\underline{f}$ ... value of best known solution, initially $\underline{f} = -\infty$
1. if $M = \emptyset$ STOP, else choose $P \in M$, $M \leftarrow M \setminus \{P\}$
2. compute upper bound $\bar{f}(P)$.
3. if $\bar{f}(P) < \underline{f}$ (P contains no OS), goto 1.
4. compute feasible solutions $\hat{f}(P)$ for P (lower bound).
5. if $\hat{f}(P) > \underline{f}$ (new best solution), put $\underline{f} \leftarrow \hat{f}(P)$
6. if $\bar{f}(P) = \hat{\underline{f}}(P)$ (no better solution in P), goto 1.
7. split P into "smaller" subproblem P_i, $M \leftarrow M \cup \{P_1, \ldots, P_k\}$
8. goto 1.

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: $\max\ c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: $\max\ c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: $\max\ c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C+\frac{8}{9}A=34$
LB: $C+D+F=30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

1 $\leftarrow$ x_A $\rightarrow$ 0

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
UB: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
UB $< 30 \Rightarrow$ no OS $\square$

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
UB: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
UB $< 30 \Rightarrow$ no OS $\square$

P_3: $x_A = 0$
UB: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
LB: $C + D + F = 30$

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: max $c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

1 $\leftarrow$ x_A $\rightarrow$ 0

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
UB: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
UB $< 30 \Rightarrow$ no OS $\square$

P_3: $x_A = 0$
UB: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
LB: $C + D + F = 30$

1 $\leftarrow$ x_E $\rightarrow$ 0

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: $\max\ c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
UB: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
UB $< 30 \Rightarrow$ no OS $\square$

P_3: $x_A = 0$
UB: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
LB: $C + D + F = 30$

$1 \quad \leftarrow \quad x_E \quad \rightarrow \quad 0$

P_4: $x_A = 0,\ x_E = 1$
UB: $E + C + D = 31$
LB: $E + C + D = 31$ $\square$

Example: $\{0,1\}$-Knapsack problem

Item	A	B	C	D	E	F	capacity
weight (a)	9	7	6	4	4	3	14
profit (c)	18	6	18	7	6	5	

sorted profit/weight: $C > A > D > F > E > B$.

upper bound: $\max\ c^T x$ s.t. $a^T x \leq 14,\ x \in [0,1]^6$ [LP-relaxation]

lower bound: sort by profit/weight and fill in this sequence

P_1: original problem
UB: $C + \frac{8}{9}A = 34$
LB: $C + D + F = 30$

$1 \quad \leftarrow \quad x_A \quad \rightarrow \quad 0$

P_2: $x_A = 1 \Rightarrow x_B = x_C = 0$
UB: $A + D + \frac{1}{3}F = 26\frac{2}{3}$
UB $< 30 \Rightarrow$ no OS $\square$

P_3: $x_A = 0$
UB: $C + D + F + \frac{1}{4}E = 31\frac{1}{2}$
LB: $C + D + F = 30$

$1 \quad \leftarrow \quad x_E \quad \rightarrow \quad 0$

P_4: $x_A = 0,\ x_E = 1$
UB: $E + C + D = 31$
LB: $E + C + D = 31$ $\square$

P_5: $x_A = x_E = 0$
UB: $C + D + F + \frac{1}{7}B = 30\frac{6}{7}$
LB $< 31 \Rightarrow$ no OS $\square$

The branch&bound tree will get huge whenever many solutions are almost optimal.

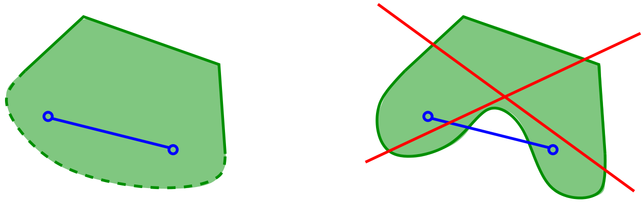
For successful branch&bound we need to answer

How can we obtain good upper and lower bounds?

Contents

1.7 Convex Sets and Convex Hull

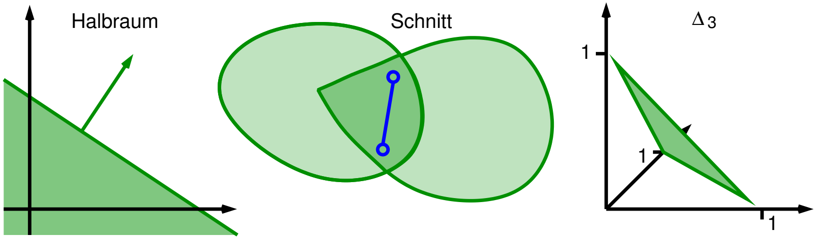
A set $C \subseteq \mathbb{R}^n$ is convex, if for all $x, y \in C$ the straight line segment $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ lies in C.



1.7 Convex Sets and Convex Hull

A set $C \subseteq \mathbb{R}^n$ is convex, if for all $x, y \in C$ the straight line segment $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0, 1]\}$ lies in C.

Examples: $\emptyset$, $\mathbb{R}^n$, halfspaces, the intersection of convex sets is convex, polyhedra, the k-dim. unit simplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

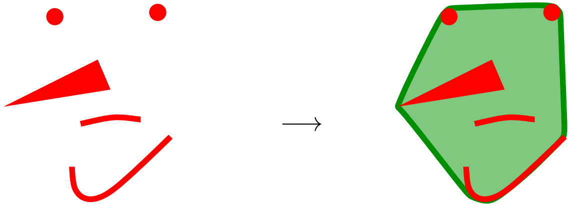


1.7 Convex Sets and Convex Hull

A set $C \subseteq \mathbb{R}^n$ is convex, if for all $x, y \in C$ the straight line segment $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ lies in C.

Examples: $\emptyset$, $\mathbb{R}^n$, halfspaces, the intersection of convex sets is convex, polyhedra, the k-dim. unit simplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

For $S \subseteq \mathbb{R}^n$ the convex hull is the intersection of all convex sets that contain S, $\operatorname{conv} S := \bigcap\{C \text{ convex} : S \subseteq C\}$.



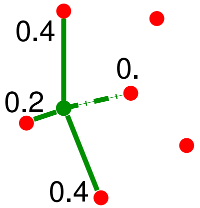
1.7 Convex Sets and Convex Hull

A set $C \subseteq \mathbb{R}^n$ is convex, if for all $x, y \in C$ the straight line segment $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ lies in C.

Examples: $\emptyset$, $\mathbb{R}^n$, halfspaces, the intersection of convex sets is convex, polyhedra, the k-dim. unit simplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

For $S \subseteq \mathbb{R}^n$ the convex hull is the intersection of all convex sets that contain S, $\operatorname{conv} S := \bigcap \{C \text{ convex} : S \subseteq C\}$.

For given points $x^{(i)} \in \mathbb{R}^n$, $i \in \{1, \dots, k\}$ and $\alpha \in \Delta_k$, the point $x = \sum \alpha_i x^{(i)}$ is a convex combination of the $x^{(i)}$.



1.7 Convex Sets and Convex Hull

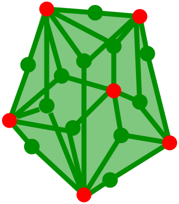
A set $C \subseteq \mathbb{R}^n$ is convex, if for all $x, y \in C$ the straight line segment $[x, y] := \{\alpha x + (1-\alpha)y : \alpha \in [0,1]\}$ lies in C.

Examples: $\emptyset$, $\mathbb{R}^n$, halfspaces, the intersection of convex sets is convex, polyhedra, the k-dim. unit simplex $\Delta_k := \{\alpha \in \mathbb{R}^k_+ : \sum_{i=1}^k \alpha_i = 1\}$

For $S \subseteq \mathbb{R}^n$ the convex hull is the intersection of all convex sets that contain S, $\operatorname{conv} S := \bigcap\{C \text{ convex} : S \subseteq C\}$.

For given points $x^{(i)} \in \mathbb{R}^n$, $i \in \{1, \dots, k\}$ and $\alpha \in \Delta_k$, the point $x = \sum \alpha_i x^{(i)}$ is a convex combination of the $x^{(i)}$.

$\operatorname{conv} S$ is the set of all convex combinations of finitely many points in S, $\operatorname{conv} S = \{\sum_{i=1}^k \alpha_i x^{(i)} : x^{(i)} \in S, i = 1, \dots, k \in \mathbb{N}, \alpha \in \Delta_k\}$.



Convex Hull and Integer Programming

Theorem

The convex hull of finitely many points is a (bounded) polyhedron.

Convex Hull and Integer Programming

Theorem

The convex hull of finitely many points is a (bounded) polyhedron.

The integer hull of a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is the convex hull of the integer points in P, $P_I := \mathrm{conv}(P \cap \mathbb{Z}^n)$.

Theorem

If $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, the integer hull P_I of polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is itself a polyhedron.

Convex Hull and Integer Programming

Theorem

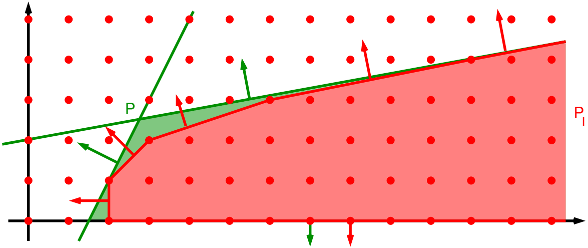
The convex hull of finitely many points is a (bounded) polyhedron.

The integer hull of a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is the convex hull of the integer points in P, $P_I := \operatorname{conv}(P \cap \mathbb{Z}^n)$.

Theorem

If $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, the integer hull P_I of polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is itself a polyhedron.

difficulty: the description of P_I is mostly unknown or excessive!



[exception e.g. for A tot. unimod., $b \in \mathbb{Z}^n$, then $P = P_I$]

Convex Hull and Integer Programming

Theorem

The convex hull of finitely many points is a (bounded) polyhedron.

The integer hull of a polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is the convex hull of the integer points in P, $P_I := \operatorname{conv}(P \cap \mathbb{Z}^n)$.

Theorem

If $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$, the integer hull P_I of polyhedron $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is itself a polyhedron.

---

difficulty: the description of P_I is mostly unknown or excessive!

If the integer hull can be given explicitly, the integer optimization problem can be solved by the simplex method:

Theorem

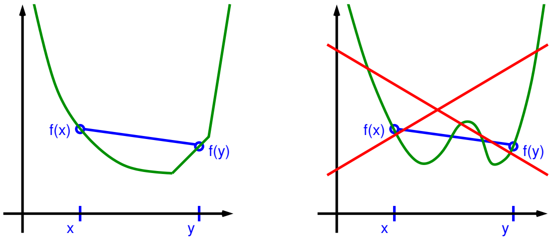
Suppose the integer hull of $P = \{x \in \mathbb{R}^n : Ax \leq b\}$ is given by $P_I = \{x \in \mathbb{R}^n : A_I x \leq b_I\}$, then:

$$\begin{aligned} \sup\{c^T x : A_I x \leq b_I, x \in \mathbb{R}^n\} &= \sup\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\}, \\ \operatorname{Argmin}\{c^T x : A_I x \leq b_I, x \in \mathbb{R}^n\} &= \operatorname{conv} \operatorname{Argmin}\{c^T x : Ax \leq b, x \in \mathbb{Z}^n\}. \end{aligned}$$

Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if $f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha)f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ for $x, y \in \mathbb{R}^n$, $x \neq y$ and $\alpha \in (0,1)$.



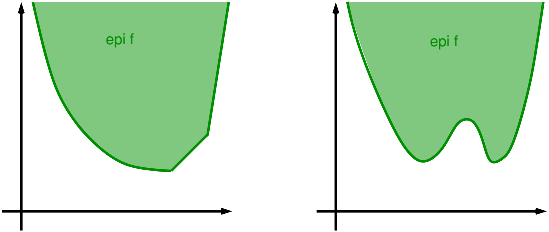
Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if
$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$, $x \neq y$ and $\alpha \in (0,1)$.

The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the set

$$\text{epi}\, f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [the points "above" $f(x)$]



Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

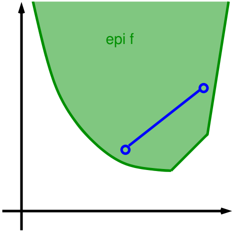
f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$, $x \neq y$ and $\alpha \in (0,1)$.

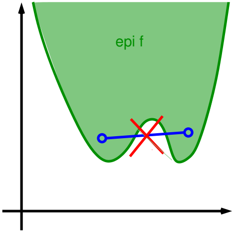
The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the set

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [the points “above” $f(x)$]

Theorem

A function is convex if and only if its epigraph is a convex set.





Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$ for $x, y \in \mathbb{R}^n$, $x \neq y$ and $\alpha \in (0,1)$.

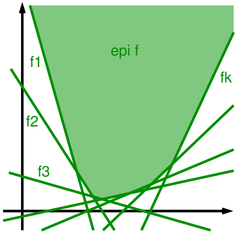
The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the set

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [the points "above" $f(x)$]

Theorem

A function is convex if and only if its epigraph is a convex set.

Ex.: for convex $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ also $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$ is convex.



Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if $f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$, $x \neq y$ and $\alpha \in (0,1)$.

The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the set

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \geq f(x) \right\}$$ [the points "above" $f(x)$]

Theorem

A function is convex if and only if its epigraph is a convex set.

Ex.: for convex $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ also $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$ is convex.

Theorem

Each local minimum of a convex function is also a global minimum, and for strictly convex functions it is unique (if it exists).

For convex functions there exist rather good optimization methods.

Convex Functions

A function $f : \mathbb{R}^n \to \bar{\mathbb{R}} := \mathbb{R} \cup \{\infty\}$ is convex if $f(\alpha x + (1-\alpha)y) \le \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$ and $\alpha \in [0,1]$.

f is strictly convex, if $f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha) f(y)$ for $x, y \in \mathbb{R}^n$, $x \ne y$ and $\alpha \in (0,1)$.

The epigraph of a function $f : \mathbb{R}^n \to \bar{\mathbb{R}}$ is the set

$$\operatorname{epi} f := \left\{ \begin{pmatrix} x \\ r \end{pmatrix} : x \in \mathbb{R}^n, r \ge f(x) \right\}$$ [the points "above" $f(x)$]

Theorem

A function is convex if and only if its epigraph is a convex set.

Ex.: for convex $f_i : \mathbb{R}^n \to \bar{\mathbb{R}}$ also $f(x) := \sup_i f_i(x)$, $x \in \mathbb{R}^n$ is convex.

Theorem

Each local minimum of a convex function is also a global minimum, and for strictly convex functions it is unique (if it exists).

For convex functions there exist rather good optimization methods.

A function f is concave, if $-f$ is convex.
(Each local maximum of a concave function is a global one.)

Contents

1.8 Relaxation

Concept applicable to arbitrary optimization problems (here maximize):

Definition

Given two optimization problems with $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ *and* $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max\ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{and} \qquad (RP) \quad \max\ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) *is a* ***relaxation*** *of* (OP) *if*

1. $\mathcal{X} \subseteq \mathcal{W}$,
2. $f(x) \leq f'(x)$ *for all* $x \in \mathcal{X}$.

1.8 Relaxation

Concept applicable to arbitrary optimization problems (here maximize):

Definition

Given two optimization problems with $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ *and* $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max\ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{and} \qquad (RP) \quad \max\ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) *is a* *relaxation* *of* (OP) *if*

1. $\mathcal{X} \subseteq \mathcal{W}$,
2. $f(x) \leq f'(x)$ *for all* $x \in \mathcal{X}$.

Let (RP) be a relaxation of (OP).

Observation

1. $v(RP) \geq v(OP)$. *[(RP) yields an upper bound]*
2. *If* (RP) *is infeasible, then so is* (OP),
3. *If* x^* *is OS of (RP) and* $x^* \in \mathcal{X}$ *with* $f'(x^*) = f(x^*)$, *then* x^* *is OS of (OP).*

1.8 Relaxation

Concept applicable to arbitrary optimization problems (here maximize):

Definition

Given two optimization problems with $\mathcal{X}, \mathcal{W} \subseteq \mathbb{R}^n$ and $f, f' : \mathbb{R}^n \to \mathbb{R}$

$$(OP) \quad \max\ f(x) \text{ s.t. } x \in \mathcal{X} \qquad \textit{and} \qquad (RP) \quad \max\ f'(x) \text{ s.t. } x \in \mathcal{W},$$

(RP) is a relaxation of (OP) if

1. *$\mathcal{X} \subseteq \mathcal{W}$,*
2. *$f(x) \leq f'(x)$ for all $x \in \mathcal{X}$.*

Let (RP) be a relaxation of (OP).

Observation

1. *$v(RP) \geq v(OP)$.* *[(RP) yields an upper bound]*
2. *If (RP) is infeasible, then so is (OP),*
3. *If x^* is OS of (RP) and $x^* \in \mathcal{X}$ with $f'(x^*) = f(x^*)$, then x^* is OS of (OP).*

Search for a suitable "small" $\mathcal{W} \supseteq \mathcal{X}$ and $f' \geq f$ so that (RP) is efficiently solvable.

A relaxation (RP) of (OP) is called exact if $v(OP) = v(RP)$.

Convex Relaxation

If convexity is required for $\mathcal{W}$ and (for max) concavity for f', this yields a convex relaxation. Mostly (but not always!) convexity ensures reasonable algorithmic solvability of the relaxation.

Convex Relaxation

If convexity is required for $\mathcal{W}$ and (for max) concavity for f', this yields a convex relaxation. Mostly (but not always!) convexity ensures reasonable algorithmic solvability of the relaxation.

Ex.: for a combinatorial max problem with finite ground set Ω, feasible solutions $\mathcal{F} \subseteq 2^{\Omega}$ and linear cost function $c \in \mathbb{R}^{\Omega}$

$$\max c^T x \text{ s.t. } x \in \operatorname{conv}\{\chi_{\Omega}(F) : F \in \mathcal{F}\}$$

is an exact convex (even linear) relaxation, but it is useful only if the polyhedron $\operatorname{conv}\{\chi(F) : F \in \mathcal{F}\}$ can be described by a reasonable inequality system $Ax \leq b$.

Convex Relaxation

If convexity is required for $\mathcal{W}$ and (for max) concavity for f', this yields a convex relaxation. Mostly (but not always!) convexity ensures reasonable algorithmic solvability of the relaxation.

Ex.: for a combinatorial max problem with finite ground set Ω, feasible solutions $\mathcal{F} \subseteq 2^\Omega$ and linear cost function $c \in \mathbb{R}^\Omega$

$$\max c^T x \text{ s.t. } x \in \operatorname{conv}\{\chi_\Omega(F) : F \in \mathcal{F}\}$$

is an exact convex (even linear) relaxation, but it is useful only if the polyhedron $\operatorname{conv}\{\chi(F) : F \in \mathcal{F}\}$ can be described by a reasonable inequality system $Ax \le b$.

In global optimization, nonlinear functions are approximated from below by convex functions on domain subdivisions.

Ex.: Consider $(OP)\ \min f(x) := \frac{1}{2}x^T Q x + q^T x \text{ s.t. } x \in [0,1]^n$
with f not convex, i.e., $\lambda_{\min}(Q) < 0$. [$\lambda_{\min}$... minimal eigenvalue]
$Q - \lambda_{\min}(Q)I$ is positive semidefinite and by $x_i^2 \le x_i$ on $[0,1]^n$ there holds
$f'(x) := \frac{1}{2}x^T(Q - \lambda_{\min}(Q)I)x + (q + \lambda_{\min}(Q)\mathbf{1})^T x \ \le\ f(x) \quad \forall x \in [0,1]^n$.
Thus, $(RP)\ \min\ f'(x) \text{ s.t. } x \in [0,1]^n$ is a convex relaxation of (OP).

LP-Relaxation for Integer Programs

For an integer program $\max\ c^T x$ s.t. $Ax \le b,\ x \in \mathbb{Z}^n$ dropping the integrality constraints yields the LP-relaxation

$$\max\ c^T x \text{ s.t. } Ax \le b,\ x \in \mathbb{R}^n.$$

It is a relaxation, because

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \le b\} \ \subseteq\ \{x \in \mathbb{R}^n : Ax \le b\} =: \mathcal{W}.$$

[basis of all standard solvers for mixed integer programs]

LP-Relaxation for Integer Programs

For an integer program $\max\ c^T x$ s.t. $Ax \leq b,\ x \in \mathbb{Z}^n$ dropping the integrality constraints yields the LP-relaxation

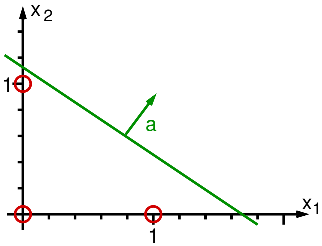
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

It is a relaxation, because

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \subseteq \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[basis of all standard solvers for mixed integer programs]

Ex.: knapsack problem: $n = 2$, weights $a = (6, 8)^T$, capacity $b = 10$,
$\max\ c^T x$ s.t. $a^T x \leq b,\ x \in \mathbb{Z}^n_+ \quad \rightarrow \quad \max\ c^T x$ s.t. $a^T x \leq b,\ x \geq 0$,



feasible integer points: ◯

LP-Relaxation for Integer Programs

For an integer program $\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{Z}^n$ dropping the integrality constraints yields the LP-relaxation

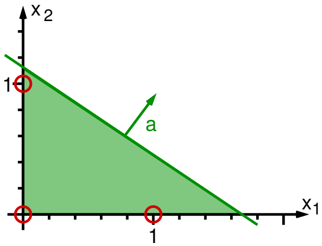
$$\max\ c^T x \text{ s.t. } Ax \leq b,\ x \in \mathbb{R}^n.$$

It is a relaxation, because

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \leq b\} \ \subseteq\ \{x \in \mathbb{R}^n : Ax \leq b\} =: \mathcal{W}.$$

[basis of all standard solvers for mixed integer programs]

Ex.: knapsack problem: $n = 2$, weights $a = (6, 8)^T$, capacity $b = 10$,
$\max\ c^T x \text{ s.t. } a^T x \leq b,\ x \in \mathbb{Z}^n_+ \quad \rightarrow \quad \max\ c^T x \text{ s.t. } a^T x \leq b,\ x \geq 0,$



feasible integer points: ◯

LP-relaxation: green

LP-Relaxation for Integer Programs

For an integer program $\max\ c^T x$ s.t. $Ax \le b,\ x \in \mathbb{Z}^n$ dropping the integrality constraints yields the LP-relaxation

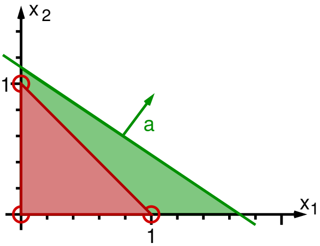
$$\max\ c^T x \text{ s.t. } Ax \le b,\ x \in \mathbb{R}^n.$$

It is a relaxation, because

$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \le b\} \ \subseteq\ \{x \in \mathbb{R}^n : Ax \le b\} =: \mathcal{W}.$$

[basis of all standard solvers for mixed integer programs]

Ex.: knapsack problem: $n = 2$, weights $a = (6, 8)^T$, capacity $b = 10$, $\max\ c^T x$ s.t. $a^T x \le b,\ x \in \mathbb{Z}^n_+ \quad \to \quad \max\ c^T x$ s.t. $a^T x \le b,\ x \ge 0$,



feasible integer points: ◯

LP-relaxation: green

best relaxation: the convex hull

LP-Relaxation for Integer Programs

For an integer program max $c^T x$ s.t. $Ax \le b,\ x \in \mathbb{Z}^n$ dropping the integrality constraints yields the LP-relaxation

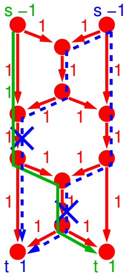
$$\max\ c^T x \text{ s.t. } Ax \le b,\ x \in \mathbb{R}^n.$$

It is a relaxation, because

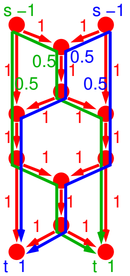
$$\mathcal{X} := \{x \in \mathbb{Z}^n : Ax \le b\} \ \subseteq\ \{x \in \mathbb{R}^n : Ax \le b\} =: \mathcal{W}.$$

[basis of all standard solvers for mixed integer programs]

Ex: integer multi-commodity flow → fractional multi-commodity flow



infeasible, $\mathcal{X} = \emptyset$



frac. feasible, $\mathcal{W} \neq \emptyset$

too large, would need convex hull

Lagrangian Relaxation

[Appl. to constrained optim. in general, here only for ineq.-constraints]

Inconvenient constraints are lifted into the cost function via a Lagrange multiplier that penalizes violations ($g : \mathbb{R}^n \to \mathbb{R}^k$):

$$(OP) \quad \begin{array}{ll} \max & f(x) \\ \text{s.t.} & g(x) \leq 0 \quad \mid \cdot \lambda \geq 0 \\ & x \in \Omega \end{array} \quad \rightarrow \quad (RP_\lambda) \quad \begin{array}{ll} \max & c^T x - \lambda^T g(x) \\ \text{s.t.} & x \in \Omega \end{array}$$

is a relaxation: $\mathcal{X} := \{x \in \Omega : g(x) \leq 0\} \subseteq \{x \in \Omega\} =: \mathcal{W}$ and for $x \in \mathcal{X}$, $\lambda \geq 0$ there holds $f(x) \leq f(x) - \lambda^T g(x) =: f'(x)$ by $g(x) \leq 0$.

Lagrangian Relaxation

[Appl. to constrained optim. in general, here only for ineq.-constraints]

Inconvenient constraints are lifted into the cost function via a Lagrange multiplier that penalizes violations ($g : \mathbb{R}^n \to \mathbb{R}^k$):

$$(OP)\quad \begin{array}{ll} \max & f(x) \\ \text{s.t.} & g(x) \le 0 \quad | \cdot \lambda \ge 0 \\ & x \in \Omega \end{array} \quad \to \quad (RP_\lambda)\quad \begin{array}{ll} \max & c^T x - \lambda^T g(x) \\ \text{s.t.} & x \in \Omega \end{array}$$

is a relaxation: $\mathcal{X} := \{x \in \Omega : g(x) \le 0\} \subseteq \{x \in \Omega\} =: \mathcal{W}$ and for $x \in \mathcal{X}$, $\lambda \ge 0$ there holds $f(x) \le f(x) - \lambda^T g(x) =: f'(x)$ by $g(x) \le 0$.

Define the dual function $\varphi(\lambda) := \sup_{x \in \Omega} \left[f(x) - g(x)^T \lambda \right] = v(RP_\lambda)$

[for each fixed x linear in λ]

- for each $\lambda \ge 0$ there holds $\varphi(\lambda) \ge v(OP)$ [upper bound]
- $\varphi(\lambda)$ easy to compute if (RP_λ) is “easy” to compute
- φ is convex, because sup of linear functions in λ
- best bound is $\inf\{\varphi(\lambda) : \lambda \ge 0\}$ [convex problem!]

well suited for convex optimization methods!

Example: Integer Multi-Commodity Flow

Let A be the node-arc incidence matrix to $D=(V,E)$, 2 goods, relax the coupling capacity constraints by $\lambda \geq 0$:

$$\begin{array}{llcccl}
 & \min\; {c^{(1)}}^T x^{(1)} & + & {c^{(2)}}^T x^{(2)} & & \\
 & \text{s.t.}\quad A x^{(1)} & & & = & b^{(1)} \\
 & & & A x^{(2)} & = & b^{(2)} \\
\lambda \cdot | & x^{(1)} & + & x^{(2)} & \leq & w \\
 & x^{(1)} \leq w, & & x^{(2)} \leq w & & \\
 & x^{(1)} \in \mathbb{Z}^E_+, & & x^{(2)} \in \mathbb{Z}^E_+. & &
\end{array}
\quad \rightarrow \quad
\begin{array}{lllr}
\min & (c^{(1)}+\lambda)^T x^{(1)} + & (c^{(2)}+\lambda)^T x^{(2)} & -\lambda^T w \\
\text{s.t.} & A x^{(1)} & & = b^{(1)} \\
 & & A x^{(2)} & = b^{(2)} \\
 & x^{(1)} \leq w, & x^{(2)} \leq w, & \\
 & x^{(1)} \in \mathbb{Z}^E_+, & x^{(2)} \in \mathbb{Z}^E_+. &
\end{array}$$

The relaxation consists of two independent min-cost-flow problems

$$(RP^{(i)}_\lambda) \quad \min\,(c^{(i)}+\lambda)^T x^{(i)} \quad \text{s.t.} \quad A x^{(i)} = b^{(i)},\; w \geq x^{(i)} \in \mathbb{Z}^E_+ \quad i \in \{1,2\}$$

These can be solved integrally and efficiently!

Example: Integer Multi-Commodity Flow

Let A be the node-arc incidence matrix to $D = (V, E)$, 2 goods, relax the coupling capacity constraints by $\lambda \geq 0$:

$$\begin{array}{lllll} & \min c^{(1)\,T}x^{(1)} & + & c^{(2)\,T}x^{(2)} & \\ & \text{s.t.} \quad Ax^{(1)} & & & = b^{(1)} \\ & & & Ax^{(2)} & = b^{(2)} \\ \lambda \cdot | & x^{(1)} & + & x^{(2)} & \leq w \\ & x^{(1)} \leq w, & & x^{(2)} \leq w & \\ & x^{(1)} \in \mathbb{Z}_+^E, & & x^{(2)} \in \mathbb{Z}_+^E. & \end{array} \rightarrow \begin{array}{lllll} \min & (c^{(1)}+\lambda)^T x^{(1)} & + (c^{(2)}+\lambda)^T x^{(2)} & - \lambda^T w \\ \text{s.t.} & Ax^{(1)} & & = b^{(1)} \\ & & Ax^{(2)} & = b^{(2)} \\ & x^{(1)} \leq w, & x^{(2)} \leq w, & \\ & x^{(1)} \in \mathbb{Z}_+^E, & x^{(2)} \in \mathbb{Z}_+^E. & \end{array}$$

The relaxation consists of two independent min-cost-flow problems

$$(RP_\lambda^{(i)}) \quad \min\, (c^{(i)}+\lambda)^T x^{(i)} \quad \text{s.t.} \quad Ax^{(i)} = b^{(i)},\ w \geq x^{(i)} \in \mathbb{Z}_+^E \quad i \in \{1, 2\}$$

These can be solved integrally and efficiently!

If Lagrangian relaxation splits the problem into independent subproblems, this is sometimes called Lagrangian decomposition. Frequently this allows to solve much bigger problems efficiently.

Does this also yield better bounds?

Comparison of Lagrange- and LP-Relaxation

Given finite $\Omega \subset \mathbb{Z}^n$ and $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll} \max & c^T x \\ \text{s.t.} & Dx \le d \quad | \cdot \lambda \ge 0 \\ & x \in \Omega \end{array} \qquad \rightarrow \qquad (RP_\lambda)\quad \begin{array}{ll} \max & c^T x + \lambda^T (d - Dx) \\ \text{s.t.} & x \in \Omega \end{array}$$

In the ex.: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ with $\Omega^{(i)} = \{x \in \mathbb{Z}_+^E : Ax = b^{(i)}, x \le w\}$, $i \in \{1, 2\}$.

Theorem

$$\inf_{\lambda \ge 0} v(RP_\lambda) = \sup\{c^T x : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

If $\operatorname{conv}\Omega$ is identical to the feasible set of the LP-relaxation of Ω, the values of the best Lagrange relaxation and the LP-relaxation match!

Comparison of Lagrange- and LP-Relaxation

Given finite $\Omega \subset \mathbb{Z}^n$ and $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll} \max & c^T x \\ \text{s.t.} & Dx \le d \quad |\cdot \lambda \ge 0 \\ & x \in \Omega \end{array} \quad \rightarrow \quad (RP_\lambda)\quad \begin{array}{ll} \max & c^T x + \lambda^T (d - Dx) \\ \text{s.t.} & x \in \Omega \end{array}$$

In the ex.: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ with $\Omega^{(i)} = \{x \in \mathbb{Z}^E_+ : Ax = b^{(i)}, x \le w\}$, $i \in \{1, 2\}$.

Theorem

$$\inf_{\lambda \ge 0} v(RP_\lambda) = \sup\{c^T x : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

If $\operatorname{conv}\Omega$ is identical to the feasible set of the LP-relaxation of Ω, the values of the best Lagrange relaxation and the LP-relaxation match!

In the ex. A is totally unimodular, thus for $i \in \{1, 2\}$ and $w \in \mathbb{Z}^E$

$$\operatorname{conv}\{x \in \mathbb{Z}^E_+ : Ax = b^{(i)}, x \le w\} = \{x \ge 0 : Ax = b^{(i)}, x \le w\}.$$

The best λ yields the value of the fractional multi-comm.-flow problem!

Comparison of Lagrange- and LP-Relaxation

Given finite $\Omega \subset \mathbb{Z}^n$ and $D \in \mathbb{Q}^{k\times n}$, $d \in \mathbb{Q}^k$,

$$(OP)\quad \begin{array}{ll} \max & c^Tx \\ \text{s.t.} & Dx \le d \quad | \cdot \lambda \ge 0 \\ & x \in \Omega \end{array} \quad \to \quad (RP_\lambda)\quad \begin{array}{ll} \max & c^Tx + \lambda^T(d - Dx) \\ \text{s.t.} & x \in \Omega \end{array}$$

In the ex.: $\Omega = \Omega^{(1)} \times \Omega^{(2)}$ with $\Omega^{(i)} = \{x \in \mathbb{Z}^E_+ : Ax = b^{(i)}, x \le w\}$, $i \in \{1, 2\}$.

Theorem

$$\inf_{\lambda\ge 0} v(RP_\lambda) = \sup\{c^Tx : Dx \le d, x \in \operatorname{conv}\Omega\}.$$

If $\operatorname{conv}\Omega$ is identical to the feasible set of the LP-relaxation of Ω, the values of the best Lagrange relaxation and the LP-relaxation match!

In the ex. A is totally unimodular, thus for $i \in \{1, 2\}$ and $w \in \mathbb{Z}^E$

$$\operatorname{conv}\{x \in \mathbb{Z}^E_+ : Ax = b^{(i)}, x \le w\} = \{x \ge 0 : Ax = b^{(i)}, x \le w\}.$$

The best λ yields the value of the fractional multi-comm.-flow problem!

In general: Let $\{x \in \mathbb{Z}^n : Ax \le b\} = \Omega$ be a formulation of Ω. Only if $\{x \in \mathbb{R}^n : Ax \le b\} \ne \operatorname{conv}\Omega$, Lagrange relaxation may yield a better value Wert than the LP-relaxation.

Contents

1.9 Application: Traveling Salesman Problem (TSP)

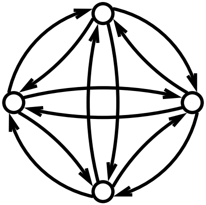
TSP: Given n cities with all pairwise distances, find a shortest round trip that visits each city exactly once.

1.9 Application: Traveling Salesman Problem (TSP)

TSP: Given n cities with all pairwise distances, find a shortest round trip that visits each city exactly once.

Comb. opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ complete digraph, costs $c \in \mathbb{R}^E$, feasible set $\mathcal{F} = \{R \subset E : R$ (dir.) cycle in $D, |R| = n\}$.
Find $R \in \text{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-complete problem
Number of tours?



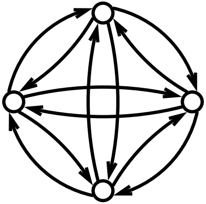
1.9 Application: Traveling Salesman Problem (TSP)

TSP: Given n cities with all pairwise distances, find a shortest round trip that visits each city exactly once.

Comb. opt.: $D = (V, E = \{(u,v) : u, v \in V, u \neq v\})$ complete digraph, costs $c \in \mathbb{R}^E$, feasible set $\mathcal{F} = \{R \subset E : R \text{ (dir.) cycle in } D, |R| = n\}$.
Find $R \in \text{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-complete problem
Number of tours? $(n-1)!$

→combinatorial explosion



$n =$ 3:	2
$n =$ 4:	$3 \cdot 2 = 6$
$n =$ 5:	$4 \cdot 3 \cdot 2 = 24$
$n =$ 10:	362880
$n =$ 11:	3628800
$n =$ 12:	39916800
$n =$ 13:	479001600
$n =$ 14:	6227020800
$n =$100:	10^{158}

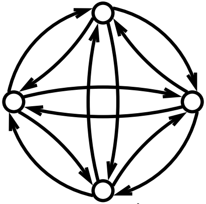
1.9 Application: Traveling Salesman Problem (TSP)

TSP: Given n cities with all pairwise distances, find a shortest round trip that visits each city exactly once.

Comb. opt.: $D = (V, E = \{(u, v) : u, v \in V, u \neq v\})$ complete digraph, costs $c \in \mathbb{R}^E$, feasible set $\mathcal{F} = \{R \subset E : R \text{ (dir.) cycle in } D, |R| = n\}$.
Find $R \in \text{Argmin}\{c(R) = \sum_{e \in R} c_e : R \in \mathcal{F}\}$

NP-complete problem
Number of tours? $(n-1)!$

→combinatorial explosion

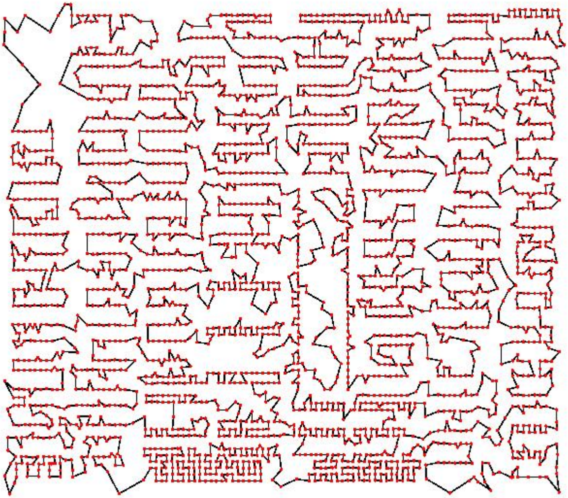


$n = 3$:	2
$n = 4$:	$3 \cdot 2 = 6$
$n = 5$:	$4 \cdot 3 \cdot 2 = 24$
$n = 10$:	362880
$n = 11$:	3628800
$n = 12$:	39916800
$n = 13$:	479001600
$n = 14$:	6227020800
$n = 100$:	10^{158}

Typical problem for:
- delivery services (parcels, pharmacy-transports)
- taxi services, breakdown services
- scheduling with setup costs [e.g. coloring cars]
- steering robots (mostly nonlinear)

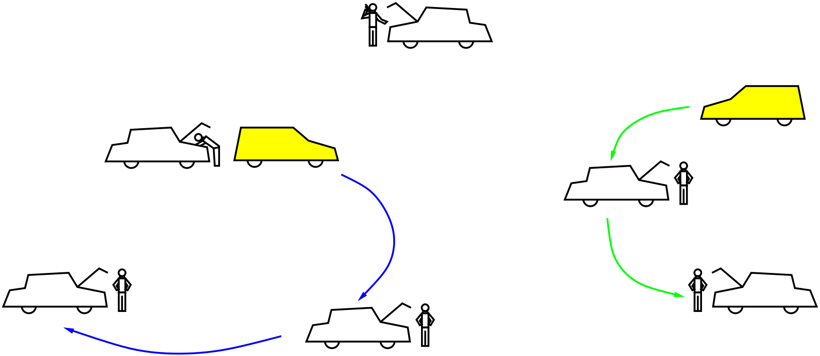
[frequently with several “vehicles” and time windows]

Drilling holes into main boards



[http://www.math.princeton.edu/tsp]

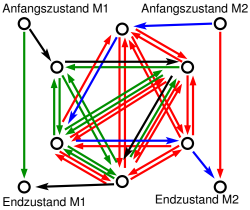
With online aspects: breakdown services

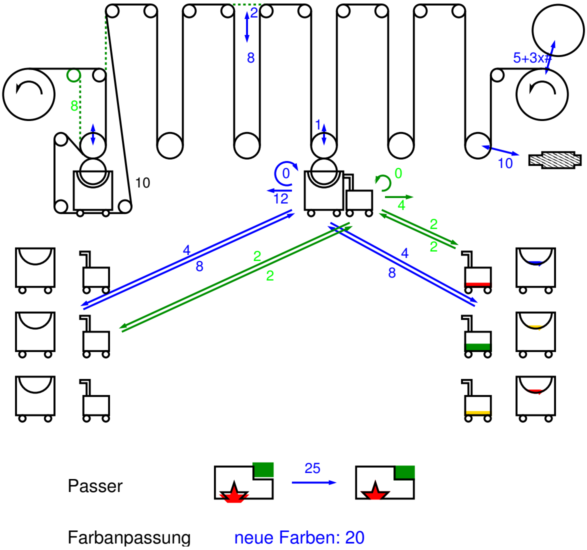


Find an assignment of cars and a sequence for each repair man, so that promised waiting periods are not exceeded.

Scheduling for long setup times

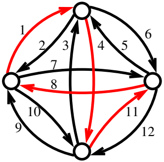
Two rotational printing machines
for printing wrapping papier



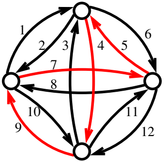


Integer Programming Model

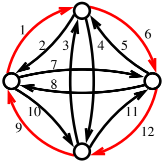
Abstract formulation uses convex hull of incidence vectors:



$R_1 = \{1, 4, 8, 11\}$



$R_2 = \{4, 5, 7, 9\}$



$R_3 = \{1, 6, 9, 12\}$

$$\chi(R_1) = \begin{bmatrix} 1\\0\\0\\1\\0\\0\\0\\1\\0\\0\\1\\0 \end{bmatrix} \qquad \chi(R_2) = \begin{bmatrix} 0\\0\\0\\1\\1\\0\\1\\0\\1\\0\\0\\0 \end{bmatrix} \qquad \chi(R_3) = \begin{bmatrix} 1\\0\\0\\0\\0\\1\\0\\0\\1\\0\\0\\1 \end{bmatrix}$$

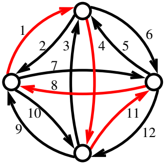
For arc lengths $c \in \mathbb{R}^E$ the length of a tour R is $\sum_{e \in R} c_e = c^T \chi(R)$.

$$(TSP) \quad \min\ c^T x \text{ s.t. } x \in \operatorname{conv}\{\chi(R) : R \text{ tour in } D = (V, E)\} =: P_{TSP}$$

Would be exact, but no linear description $A_I x \leq b_I$ of P_{TSP} is known!

Integer Formulation of (TSP)

Goal: wrap P_{TSP} by a bigger polytope $P = \{x \in \mathbb{R}^n : Ax \leq b\} \supseteq P_{TSP}$ as tightly as possible so that at least $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ tour}\}$.

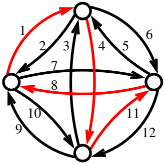


An equation/inequality is feasible for P_{TSP}, if it holds for all $x \in \{\chi(R) \colon R \text{ tour}\}$.

Suggestions?

Integer Formulation of (TSP)

Goal: wrap P_{TSP} by a bigger polytope $P = \{x \in \mathbb{R}^n : Ax \le b\} \supseteq P_{TSP}$ as tightly as possible so that at least $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ tour}\}$.



An equation/inequality is feasible for P_{TSP}, if it holds for all $x \in \{\chi(R) : R \text{ tour}\}$.

0-1 cube: $0 \le x \le 1$ is feasible

degree constraints:
exactly one arc exits and enters each node,

$$\text{for } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

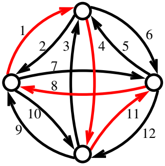
(exactly one 1 per row/column $\leftrightarrow$ assignment problem)

Is this a formulation? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ tour}\}$?

Integer Formulation of (TSP)

Goal: wrap P_{TSP} by a bigger polytope $P = \{x \in \mathbb{R}^n : Ax \le b\} \supseteq P_{TSP}$ as tightly as possible so that at least $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ tour}\}$.

An equation/inequality is feasible for P_{TSP}, if it holds for all $x \in \{\chi(R) : R \text{ tour}\}$.

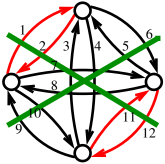


0-1 cube: $0 \le x \le 1$ is feasible

degree constraints:
exactly one arc exits and enters each node,

$$\text{for } v \in V: \quad \sum_{(v,u)\in E} x_{(v,u)} = 1, \quad \sum_{(u,v)\in E} x_{(u,v)} = 1$$

(exactly one 1 per row/column $\leftrightarrow$ assignment problem)



Is this a formulation? $P \cap \mathbb{Z}^E = \{\chi(R) : R \text{ tour}\}$?

Subtour elimination constraints:
At least one arc must exit each proper subset of nodes,

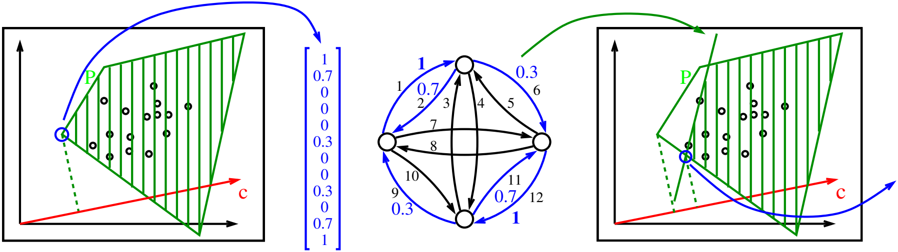
$$\text{for } S \subset V, 2 \le |S| \le n-2: \quad \sum_{e\in\delta^+(S)} x_e \ge 1$$

This is now a formulation, but it needs roughly 2^n inequalities!

Solving the TSP LP-Relaxation

Requires a cutting plane approach:
The first relaxation is the assignment problem (box+degree constr.)
Its solutions is integral and consists of distinct cycles in general.

From now on the bound is improved iteratively by subtour elim. constr.

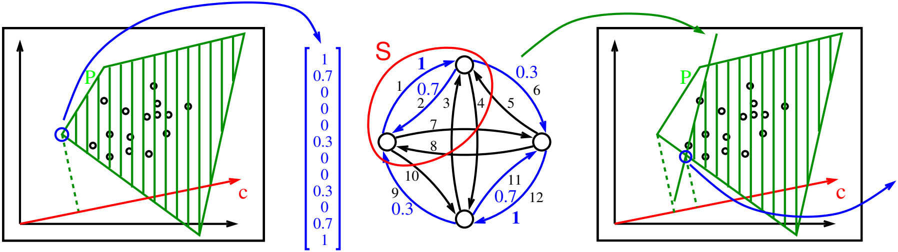


Separation problem: find $S \subset V, 2 \leq |S| \leq n-2: \quad \sum_{e \in \delta^+(S)} x_e \not\geq 1.$

Solving the TSP LP-Relaxation

Requires a cutting plane approach:
The first relaxation is the assignment problem (box+degree constr.)
Its solutions is integral and consists of distinct cycles in general.

From now on the bound is improved iteratively by subtour elim. constr.

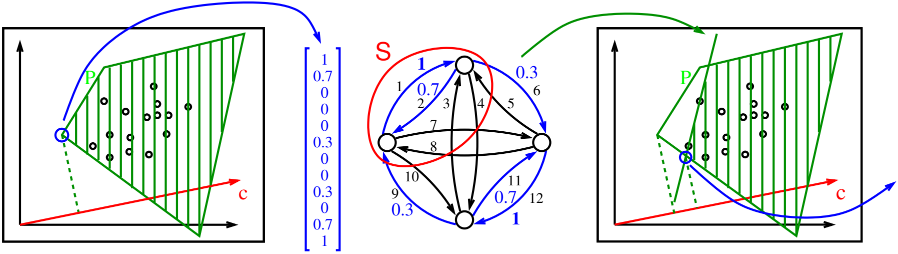


Separation problem: find $S \subset V, 2 \leq |S| \leq n-2 : \quad \sum_{e \in \delta^+(S)} x_e \not\geq 1.$
$\Leftrightarrow$ Find, in network $D = (V, E)$ with capacities x, a cut $x(\delta^+(S)) < 1$.
$\rightarrow$ Maximum s-t-flow/minimum s-t-cut for $s, t \in V$, solvable exactly!

Solving the TSP LP-Relaxation

Requires a cutting plane approach:
The first relaxation is the assignment problem (box+degree constr.)
Its solutions is integral and consists of distinct cycles in general.

From now on the bound is improved iteratively by subtour elim. constr.



Separation problem: find $S \subset V, 2 \leq |S| \leq n-2: \quad \sum_{e\in\delta^+(S)} x_e \not\geq 1.$
$\Leftrightarrow$ Find, in network $D = (V, E)$ with capacities x, a cut $x(\delta^+(S)) < 1$.
$\rightarrow$ Maximum s-t-flow/minimum s-t-cut for $s, t \in V$, solvable exactly!

---

degree+subtour yield high quality bounds, but are still far from P_{TSP}!
Bound can be improved by further ineqs (comb-, etc.),
but the solution of the relaxation almost never becomes integral!

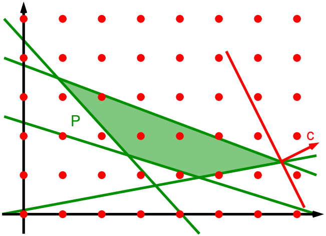
General Cutting Planes

There exist several problem independent general cutting planes, that are used in state-of-the-art solvers for integer programming:

- Gomory-cuts [rounding down coefficients by $\lfloor\cdot\rfloor$]

  if $a^T x \leq \beta$ is feasible for $x \in P \cap \mathbb{Z}^n_+$
  then by $\lfloor a \rfloor^T x \leq a^T x$
  also $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ feasible.

  For non integral OS violated inequs. of this type can be constructed.



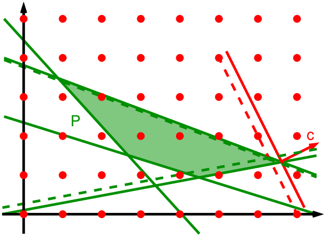
General Cutting Planes

There exist several problem independent general cutting planes, that are used in state-of-the-art solvers for integer programming:

- Gomory-cuts [rounding down coefficients by $\lfloor \cdot \rfloor$]

  if $a^T x \leq \beta$ is feasible for $x \in P \cap \mathbb{Z}_+^n$
  then by $\lfloor a \rfloor^T x \leq a^T x$
  also $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ feasible.

  For non integral OS violated inequs. of this type can be constructed.



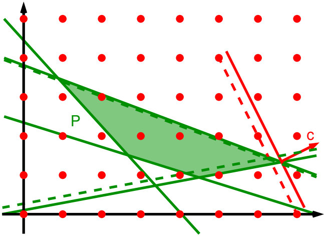
General Cutting Planes

There exist several problem independent general cutting planes, that are used in state-of-the-art solvers for integer programming:

- Gomory-cuts [rounding down coefficients by $\lfloor\cdot\rfloor$]

  if $a^T x \leq \beta$ is feasible for $x \in P \cap \mathbb{Z}^n_+$
  then by $\lfloor a \rfloor^T x \leq a^T x$
  also $\lfloor a \rfloor^T x \leq \lfloor \beta \rfloor$ feasible.

  For non integral OS violated inequs. of this type can be constructed.



- lift-and-project cuts,
- clique inequalities,

etc.

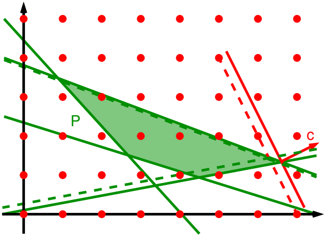
General Cutting Planes

There exist several problem independent general cutting planes, that are used in state-of-the-art solvers for integer programming:

- Gomory-cuts [rounding down coefficients by $\lfloor\cdot\rfloor$]

  if $a^T x \le \beta$ is feasible for $x \in P \cap \mathbb{Z}^n_+$
  then by $\lfloor a \rfloor^T x \le a^T x$
  also $\lfloor a \rfloor^T x \le \lfloor \beta \rfloor$ feasible.

  For non integral OS violated inequs. of this type can be constructed.



- lift-and-project cuts,
- clique inequalities,

etc.

LP-relaxation with cutting planes gives rise to good bounds (upper for maximization, lower for minimization problems), the solutions of the relaxation are (almost) never integral, but are often close to integer solutions of good quality.

Contents

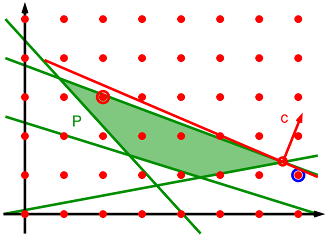
1.10 Finding "Good" Solutions, Heuristics

[Heuristic has greek origin find/invent]

For "small" $x \in \mathbb{Z}^n$ standard rounding of LP solutions typically yields infeasible or bad solutions (even if the bound is good).

It may happen that no feasible point is in the neighborhood of the LP solution!

State-of-the-art solvers employ sophisticated general purpose rounding heuristics (e.g., feasibility pump).



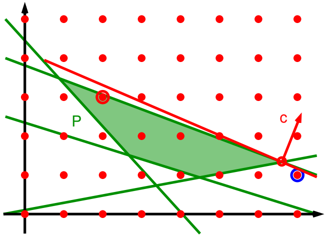
1.10 Finding “Good” Solutions, Heuristics

[Heuristic has greek origin find/invent]

For “small” $x \in \mathbb{Z}^n$ standard rounding of LP solutions typically yields infeasible or bad solutions (even if the bound is good).

It may happen that no feasible point is in the neighborhood of the LP solution!

State-of-the-art solvers employ sophisticated general purpose rounding heuristics (e.g., feasibility pump).



In general: Integer problems are NP-hard, have many “local optima” (no close better solutions) and likely finding the optimum requires (partial) enumeration. It may even be difficult to find any feasible solution!

$\rightarrow$ In applications one exploits problem specific knowledge!

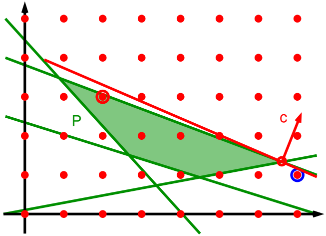
1.10 Finding "Good" Solutions, Heuristics

[Heuristic has greek origin find/invent]

For "small" $x \in \mathbb{Z}^n$ standard rounding of LP solutions typically yields infeasible or bad solutions (even if the bound is good).

It may happen that no feasible point is in the neighborhood of the LP solution!

State-of-the-art solvers employ sophisticated general purpose rounding heuristics (e.g., feasibility pump).



In general: Integer problems are *NP*-hard, have many "local optima" (no close better solutions) and likely finding the optimum requires (partial) enumeration. It may even be difficult to find any feasible solution!

$\rightarrow$ In applications one exploits problem specific knowledge!

Rough algorithmic scheme:

- generate a (feasible?) starting solution (often based on LP-sol.)
- iteratively improve the solution by some local search method (locally exact, simulated annealing, tabu search, genetic algorithms, etc.)

Starting Solution based on LP-Relaxation

Typical approaches:

- Often exactly one of several $\{0,1\}$-variables has to be selected:

  LP-relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpret value of x_i as the probability that x_i has to be set to 1 and generate several such solutions randomly, select the best.

Starting Solution based on LP-Relaxation

Typical approaches:

- Often exactly one of several $\{0,1\}$-variables has to be selected:

  LP-relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpret value of x_i as the probability that x_i has to be set to 1 and generate several such solutions randomly, select the best.

- Deviations from constraints with large dual variables result in strong losses in objective value

  $\Rightarrow$ try to satisfy those in rounding.

Starting Solution based on LP-Relaxation

Typical approaches:

- Often exactly one of several $\{0,1\}$-variables has to be selected:

  LP-relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpret value of x_i as the probability that x_i has to be set to 1 and generate several such solutions randomly, select the best.
- Deviations from constraints with large dual variables result in strong losses in objective value

  $\Rightarrow$ try to satisfy those in rounding.
- Successive fixing: Set one or several variables whose value is "almost" intregral to the rounded value and resolve the LP for the remaining variables (may require back tracking if infeasible).

Starting Solution based on LP-Relaxation

Typical approaches:

- Often exactly one of several $\{0,1\}$-variables has to be selected:

  LP-relaxation: $\sum_{i \in N} x_i = 1,\ x_i \in [0,1]$

  Interpret value of x_i as the probability that x_i has to be set to 1 and generate several such solutions randomly, select the best.
- Deviations from constraints with large dual variables result in strong losses in objective value
  $\Rightarrow$ try to satisfy those in rounding.
- Successive fixing: Set one or several variables whose value is "almost" intregral to the rounded value and resolve the LP for the remaining variables (may require back tracking if infeasible).

For some basic problems there exist rounding methods that generate feasible solutions with quality guarantee from LP solutions (approximation algorithms), these are a valuable source of good ideas for designing new rounding methods.

Improvement Methods: Principles

common basic elements:

- **declare a search neighborhood:** with respect to the current solution it describes which solutions "close by" will or may be investigated (e.g. all obtainable by certain exchange operations or by freeing certain variables with local post optimization etc.)
  mathematically: each solution $\hat{x}$ is a assigned a (neighborhood-) set $\mathcal{N}(\hat{x})$ of neighboring solutions.

Improvement Methods: Principles

common basic elements:

- **declare a search neighborhood:** with respect to the current solution it describes which solutions “close by” will or may be investigated (e.g. all obtainable by certain exchange operations or by freeing certain variables with local post optimization etc.) mathematically: each solution $\hat{x}$ is a assigned a (neighborhood-) set $\mathcal{N}(\hat{x})$ of neighboring solutions.
- **define a progress measure:** a merit function $f(x)$ for newly generated solutions that combines cost function and penalties for infeasibilities
  [cmp. merit- and filter-approach in nonlin. opt.]

Improvement Methods: Principles

common basic elements:

- **declare a search neighborhood:** with respect to the current solution it describes which solutions "close by" will or may be investigated (e.g. all obtainable by certain exchange operations or by freeing certain variables with local post optimization etc.)
  mathematically: each solution $\hat{x}$ is a assigned a (neighborhood-) set $\mathcal{N}(\hat{x})$ of neighboring solutions.
- **define a progress measure:** a merit function $f(x)$ for newly generated solutions that combines cost function and penalties for infeasibilities
  [cmp. merit- and filter-approach in nonlin. opt.]
- **fix an acceptance-scheme:** serves to decide which of the new solutions will be used to continue the search; worse solutions may be accepted sometimes in order to allow leaving local optima.

Locally Exact Methods/Local Enumerartion

Define $\mathcal{N}(\cdot)$ so that $\quad (P_{\hat{x}}) \quad \min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$
is solvable exactly by a polynomial time algorithm or by complete enumeration for each $\hat{x}$.

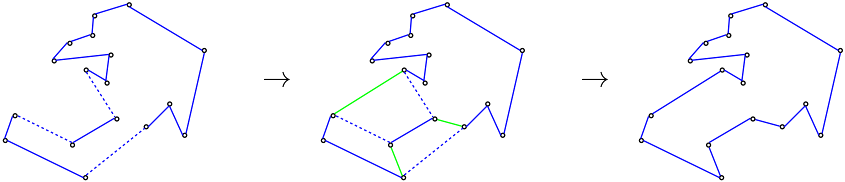
0. determine a starting solution $\hat{x}$
1. solve $(P_{\hat{x}}) \to \bar{x}$
2. if $f(\bar{x})$ is better than $f(\hat{x})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1., else STOP.

Locally Exact Methods/Local Enumerartion

Define $\mathcal{N}(\cdot)$ so that $\quad (P_{\hat{x}}) \quad \min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$
is solvable exactly by a polynomial time algorithm or by complete enumeration for each $\hat{x}$.

0. determine a starting solution $\hat{x}$
1. solve $(P_{\hat{x}}) \rightarrow \bar{x}$
2. if $f(\bar{x})$ is better than $f(\hat{x})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1., else STOP.

Ex.: 3-opt for TSP: remove 3 edges and concat parts optimally

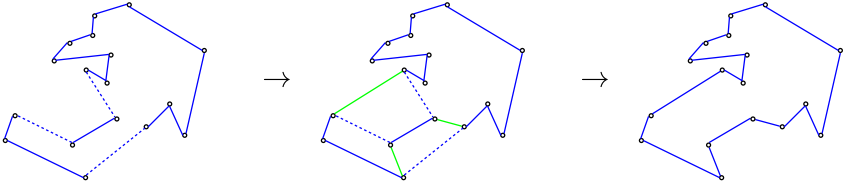


Locally Exact Methods/Local Enumerartion

Define $\mathcal{N}(\cdot)$ so that $\quad (P_{\hat{x}}) \quad \min f(x)$ s.t. $x \in \mathcal{N}(\hat{x})$
is solvable exactly by a polynomial time algorithm or by complete enumeration for each $\hat{x}$.

0. determine a starting solution $\hat{x}$
1. solve $(P_{\hat{x}}) \to \bar{x}$
2. if $f(\bar{x})$ is better than $f(\hat{x})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1., else STOP.

Ex.: 3-opt for TSP: remove 3 edges and concat parts optimally



---

the art: find a powerful and large neighborhood for which $(P_{\hat{x}})$ is still polynomially solvable.

The number of iterations may be exponential none the less!

Simulated Annealing (simulates a slow cooling process)

Select, in step k, randomly some $\bar{x}$ from $\mathcal{N}(\hat{x})$. Accept it if $f(\bar{x})$ is better than $f(\hat{x})$, otherwise accept it only with probability

$$\exp\left(\frac{-|f(\hat{x}) - f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. given a starting $\hat{x}$, fix sequence $\{T_k > 0\}_{k\in\mathbb{N}} \searrow 0$, put $k = 0$.
1. choose randomly (uniformly) $\bar{x} \in \mathcal{N}(\hat{x})$, put $k \leftarrow k + 1$
2. if $f(\bar{x})$ is better than $f(\hat{x})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1
3. draw a uniform random number $\zeta \in [0, 1]$;
   if $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1
4. goto 1 (without changing $\hat{x}$).

Choose $\mathcal{N}(\cdot)$ so that each x is reachable via intermediate steps.

Simulated Annealing (simulates a slow cooling process)

Select, in step k, randomly some $\bar{x}$ from $\mathcal{N}(\hat{x})$. Accept it if $f(\bar{x})$ is better than $f(\hat{x})$, otherwise accept it only with probability

$$\exp\left(\frac{-|f(\hat{x})-f(\bar{x})|}{T_k}\right) \qquad (0 < T_k \to 0 \text{ für } k \to \infty).$$

0. given a starting $\hat{x}$, fix sequence $\{T_k > 0\}_{k\in\mathbb{N}} \searrow 0$, put $k = 0$.
1. choose randomly (uniformly) $\bar{x} \in \mathcal{N}(\hat{x})$, put $k \leftarrow k+1$
2. if $f(\bar{x})$ is better than $f(\hat{x})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1
3. draw a uniform random number $\zeta \in [0,1]$;
   if $\zeta < \exp(\frac{-|f(\hat{x})-f(\bar{x})|}{c_k})$, put $\hat{x} \leftarrow \bar{x}$ and goto 1
4. goto 1 (without changing $\hat{x}$).

Choose $\mathcal{N}(\cdot)$ so that each x is reachable via intermediate steps.

If the (temperature-/cooling-)sequence T_k goes to zero slowly enough, each x is visited with positive probability over time (complete enumeration), therefore also the optimum (but after how long?)

Tabu-Search

Idea: try to generate highly diverse solutions

Describe $\mathcal{N}(\cdot)$ by exchange rules $r \in \mathcal{R}$ and save used rules in a tabu list $\mathcal{L}$. For any new $\bar{x}$ at least one rule $r \in \mathcal{R} \setminus \mathcal{L}$ should be used or its value must improve.

0. determine a starting $\hat{x}$, put $\mathcal{L} = \emptyset$.
1. generate several $x \in \mathcal{N}(\hat{x})$ by repeatedly applying randomly selected rules of $\mathcal{R}$, collect those in set S.
2. choose next $\bar{x}$ from S according to tabu list $\mathcal{L}$ and $f(\cdot)$.
3. update the tabu list $\mathcal{L}$, put $\hat{x} \leftarrow \bar{x}$, goto 1.

Tabu-Search

Idea: try to generate highly diverse solutions

Describe $\mathcal{N}(\cdot)$ by exchange rules $r \in \mathcal{R}$ and save used rules in a tabu list $\mathcal{L}$. For any new $\bar{x}$ at least one rule $r \in \mathcal{R} \setminus \mathcal{L}$ should be used or its value must improve.

0. determine a starting $\hat{x}$, put $\mathcal{L} = \emptyset$.
1. generate several $x \in \mathcal{N}(\hat{x})$ by repeatedly applying randomly selected rules of $\mathcal{R}$, collect those in set S.
2. choose next $\bar{x}$ from S according to tabu list $\mathcal{L}$ and $f(\cdot)$.
3. update the tabu list $\mathcal{L}$, put $\hat{x} \leftarrow \bar{x}$, goto 1.

Ex. TSP: $\mathcal{R} = \{r_{ij} :=$ switch positions of towns i and $j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij}$ was used in the last $n/10$ steps$\}$

Tabu-Search

Idea: try to generate highly diverse solutions

Describe $\mathcal{N}(\cdot)$ by exchange rules $r \in \mathcal{R}$ and save used rules in a tabu list $\mathcal{L}$. For any new $\bar{x}$ at least one rule $r \in \mathcal{R} \setminus \mathcal{L}$ should be used or its value must improve.

0. determine a starting $\hat{x}$, put $\mathcal{L} = \emptyset$.
1. generate several $x \in \mathcal{N}(\hat{x})$ by repeatedly applying randomly selected rules of $\mathcal{R}$, collect those in set S.
2. choose next $\bar{x}$ from S according to tabu list $\mathcal{L}$ and $f(\cdot)$.
3. update the tabu list $\mathcal{L}$, put $\hat{x} \leftarrow \bar{x}$, goto 1.

Ex. TSP: $\mathcal{R} = \{r_{ij} :=$ switch positions of towns i and $j\}$.
$\mathcal{L} = \{r_{ij} : r_{ij}$ was used in the last $n/10$ steps$\}$

By using rules $\mathcal{R}$ each solution should be reachable.
No general theoretical insights or quality guarantees seem to exist.

Genetic Algorithms

Idea: Let evolution work for you (and wait in the meantime).

From some population generate the next population by selection (choose next parents), recombination (exchange parts of solutions) and mutation (modify some elements randomly).

0. Choose $k \in \mathbb{N}$ and determine a starting population $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. determine the average fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. delete x from $\mathcal{P}$ with probability prop. to $\frac{f(x)}{\bar{f}}$, until $|P| = 2k$.
3. form k random pairs out of $\mathcal{P}$, generate for each pair several offsprings by recombination and mutation $\to \bar{\mathcal{P}}$
4. Put $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, goto 1.

Genetic Algorithms

Idea: Let evolution work for you (and wait in the meantime).

From some population generate the next population by selection (choose next parents), recombination (exchange parts of solutions) and mutation (modify some elements randomly).

0. Choose $k \in \mathbb{N}$ and determine a starting population $\mathcal{P}$, $|\mathcal{P}| \geq 2k$.
1. determine the average fitness $\bar{f} = \sum_{x \in \mathcal{P}} f(x)/|\mathcal{P}|$
2. delete x from $\mathcal{P}$ with probability prop. to $\frac{f(x)}{\bar{f}}$, until $|P| = 2k$.
3. form k random pairs out of $\mathcal{P}$, generate for each pair several offsprings by recombination and mutation $\rightarrow \bar{\mathcal{P}}$
4. Put $\mathcal{P} \leftarrow \bar{\mathcal{P}}$, goto 1.

- many experiments use populations of size 1 (!!!)
- theory indicates that simulated annealing is better in locating optima

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)
- Almost all hope to find optima by (slightly influencing) chance (fits to NP!).

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)
- Almost all hope to find optima by (slightly influencing) chance (fits to *NP*!).

Advantages:

- Even without understanding the problem it is easy to implement something quickly, first solutions are obtained in short time, rules are adapted easily.

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)
- Almost all hope to find optima by (slightly influencing) chance (fits to *NP*!).

Advantages:

- Even without understanding the problem it is easy to implement something quickly, first solutions are obtained in short time, rules are adapted easily.
- There are many parameters to play with.

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)
- Almost all hope to find optima by (slightly influencing) chance (fits to *NP*!).

Advantages:

- Even without understanding the problem it is easy to implement something quickly, first solutions are obtained in short time, rules are adapted easily.
- There are many parameters to play with.

Disadvantages:

- There are many parameters to adjust without any guidance!

Remarks

- SA, TS, GA are meta-heuristics (problem-independent schemes).
- meta-heuristics-industry considers arbitrary combinations yielding zillions of „new“ methods (ant-colony-, particle-swarm-, etc.)
- Almost all hope to find optima by (slightly influencing) chance (fits to *NP*!).

Advantages:

- Even without understanding the problem it is easy to implement something quickly, first solutions are obtained in short time, rules are adapted easily.
- There are many parameters to play with.

Disadvantages:

- There are many parameters to adjust without any guidance!
- “Convergence” of a method does not imply any quality guarantee. Without some related relaxation the distance to an optimal solution is entirely open (sometimes rather extreme).

Contents

1.11 Mixed-Integer Optimization (MIP)

x needs to be integral on some subset of indices $G \subseteq \{1, \ldots, n\}$.

$$\max \; c^T x \text{ s.t. } Ax \geq b, \; x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Contains integer optimization as special case but comprises much more.

1.11 Mixed-Integer Optimization (MIP)

x needs to be integral on some subset of indices $G \subseteq \{1, \ldots, n\}$.

$$\max \ c^T x \text{ s.t. } Ax \geq b, \ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Contains integer optimization as special case but comprises much more.

Example application: facility location with fixed costs

Given a set K of customers with demands b_k and a set M of potential locations for ware houses, each with opening cost c_m, capacity b_m and transportation costs c_{km} per unit for $k \in K, m \in M$. Which locations should be opened?

variables:

1.11 Mixed-Integer Optimization (MIP)

x needs to be integral on some subset of indices $G \subseteq \{1, \dots, n\}$.

$$\max \; c^T x \text{ s.t. } Ax \geq b, \; x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Contains integer optimization as special case but comprises much more.

Example application: facility location with fixed costs

Given a set K of customers with demands b_k and a set M of potential locations for ware houses, each with opening cost c_m, capacity b_m and transportation costs c_{km} per unit for $k \in K, m \in M$. Which locations should be opened?

variables:

$x_m \in \{0, 1\}$, $m \in M$ ... ware house m is opened

$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ... amount deliverd by m to customer k

constraints:

1.11 Mixed-Integer Optimization (MIP)

x needs to be integral on some subset of indices $G \subseteq \{1, \ldots, n\}$.

$$\max \; c^T x \text{ s.t. } Ax \geq b, \; x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Contains integer optimization as special case but comprises much more.

Example application: facility location with fixed costs

Given a set K of customers with demands b_k and a set M of potential locations for ware houses, each with opening cost c_m, capacity b_m and transportation costs c_{km} per unit for $k \in K, m \in M$. Which locations should be opened?

variables:
$x_m \in \{0, 1\}$, $m \in M$ ...ware house m is opened
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ...amount deliverd by m to customer k

constraints:
$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ...demand of customer k is satisfied
$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ...ware house m distributes at most b_m.

cost function:

1.11 Mixed-Integer Optimization (MIP)

x needs to be integral on some subset of indices $G \subseteq \{1, \dots, n\}$.

$$\max \ c^T x \text{ s.t. } Ax \geq b, \ x \in \mathbb{R}^n, x_G \in \mathbb{Z}^G$$

Contains integer optimization as special case but comprises much more.

Example application: facility location with fixed costs

Given a set K of customers with demands b_k and a set M of potential locations for ware houses, each with opening cost c_m, capacity b_m and transportation costs c_{km} per unit for $k \in K, m \in M$. Which locations should be opened?

variables:
$x_m \in \{0, 1\}$, $m \in M$ ...ware house m is opened
$x_{km} \in \mathbb{R}_+$, $k \in K, m \in M$ ...amount deliverd by m to customer k

constraints:
$\sum_{m \in M} x_{km} = b_k$, $k \in K$ ...demand of customer k is satisfied
$\sum_{k \in K} x_{km} \leq b_m x_m$, $m \in M$ ...ware house m distributes at most b_m.

cost function:
$\min \sum_{k \in K, m \in M} c_{km} x_{km} + \sum_{m \in M} c_m x_m = c^T x$

Modelling techniques in MIP

Conditional Inequalities: inequalities that need to hold only in dependence on certain decisions. If they need not hold, they are satisfied by adding a big M-term.

Modelling techniques in MIP

Conditional Inequalities: inequalities that need to hold only in dependence on certain decisions. If they need not hold, they are satisfied by adding a big M-term.

Example: If train A departs at time t_A before train B departs at t_B, a "headway time" $t_{AB} > 0$ has to be observed, i.e. $t_B \geq t_A + t_{AB}$.
If, on the other hand, train B departs before A, this requires $t_A \geq t_B + t_{BA}$. Of course, the later train may depart much later as well.

variables:

Modelling techniques in MIP

Conditional Inequalities: inequalities that need to hold only in dependence on certain decisions. If they need not hold, they are satisfied by adding a big M-term.

Example: If train A departs at time t_A before train B departs at t_B, a "headway time" $t_{AB} > 0$ has to be observed, i.e. $t_B \geq t_A + t_{AB}$.
If, on the other hand, train B departs before A, this requires $t_A \geq t_B + t_{BA}$. Of course, the later train may depart much later as well.

variables: ($M \gg 0$, greater than latest departure of t_A and t_B):
$x_{AB} \in \{0, 1\}$ ... 1 if A before B, 0 otherwise
$t_A, t_B \in [0, M]$... departure time

constraints:

Modelling techniques in MIP

Conditional Inequalities: inequalities that need to hold only in dependence on certain decisions. If they need not hold, they are satisfied by adding a big M-term.

Example: If train A departs at time t_A before train B departs at t_B, a "headway time" $t_{AB} > 0$ has to be observed, i.e. $t_B \geq t_A + t_{AB}$.
If, on the other hand, train B departs before A, this requires $t_A \geq t_B + t_{BA}$. Of course, the later train may depart much later as well.

variables: ($M \gg 0$, greater than latest departure of t_A and t_B):
$x_{AB} \in \{0, 1\}$ ... 1 if A before B, 0 otherwise
$t_A, t_B \in [0, M]$... departure time

constraints:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ ... only of importance if $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + M x_{AB}$ ... only of importnace if $x_{AB} = 0$.

Modelling techniques in MIP

Conditional Inequalities: inequalities that need to hold only in dependence on certain decisions. If they need not hold, they are satisfied by adding a big M-term.

Example: If train A departs at time t_A before train B departs at t_B, a "headway time" $t_{AB} > 0$ has to be observed, i.e. $t_B \geq t_A + t_{AB}$.
If, on the other hand, train B departs before A, this requires $t_A \geq t_B + t_{BA}$. Of course, the later train may depart much later as well.

variables: ($M \gg 0$, greater than latest departure of t_A and t_B):
$x_{AB} \in \{0, 1\}$ ... 1 if A before B, 0 otherwise
$t_A, t_B \in [0, M]$... departure time

constraints:
$t_A + t_{AB} \leq t_B + M(1 - x_{AB})$ ... only of importance if $x_{AB} = 1$
$t_B + t_{BA} \leq t_A + Mx_{AB}$ ... only of importnace if $x_{AB} = 0$.

- M too big $\rightarrow$ ineq. in LP-relaxation too weak $\rightarrow$ bad bound
- reasonable, if violation gap of ineq. is well controlled (see the example on facility location)
- useful in branch&bound if the decision is used for branching

Modelling Logical Constraints

For $x_i \in \{0,1\}$, $x_i = 1$ often represents "expression i is true".
Logical expressions can then be generated as follows:

log. expression	formulation
$x_2 = (\text{not } x_1)$	

Modelling Logical Constraints

For $x_i \in \{0,1\}$, $x_i = 1$ often represents "expression i is true".
Logical expressions can then be generated as follows:

log. expression	formulation
$x_2 = (\text{not } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ or } x_2)$	

Modelling Logical Constraints

For $x_i \in \{0,1\}$, $x_i = 1$ often represents "expression i is true".
Logical expressions can then be generated as follows:

log. expression	formulation
$x_2 = (\text{not } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ or } x_2)$	$x_3 \geq x_1$, $x_3 \geq x_2$, $x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ and } x_2)$	$x_3 \leq x_1$, $x_3 \leq x_2$, $x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	

Modelling Logical Constraints

For $x_i \in \{0,1\}$, $x_i = 1$ often represents "expression i is true".
Logical expressions can then be generated as follows:

log. expression	formulation
$x_2 = (\text{not } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ or } x_2)$	$x_3 \geq x_1$, $x_3 \geq x_2$, $x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ and } x_2)$	$x_3 \leq x_1$, $x_3 \leq x_2$, $x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	

Modelling Logical Constraints

For $x_i \in \{0,1\}$, $x_i = 1$ often represents "expression i is true".
Logical expressions can then be generated as follows:

log. expression	formulation
$x_2 = (\text{not } x_1)$	$x_2 = 1 - x_1$
$x_3 = (x_1 \text{ or } x_2)$	$x_3 \geq x_1,\ x_3 \geq x_2,\ x_3 \leq x_1 + x_2$
$x_3 = (x_1 \text{ and } x_2)$	$x_3 \leq x_1,\ x_3 \leq x_2,\ x_3 \geq x_1 + x_2 - 1$
$x_1 \Rightarrow x_2$	$x_1 \leq x_2$
$x_1 \Leftrightarrow x_2$	$x_1 = x_2$

Remark: Together with $0 \leq x_i \leq 1$ these constraints describe

$$\operatorname{conv}\left\{ \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \{0,1\}^3 : \text{die } x_i \text{ satisfy the logical expression} \right\}.$$

Using this technique models of further expressions can be derived.
Exercise: $x_3 = (x_1 \text{ xor } x_2)$

General Cutting Planes for MIP

Like in integer programming, “conv” is the best linear relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \leq b, x_G \in \mathbb{Z}^G\}.$$

$Ax \leq b$ is the LP-relaxation, P_G is approximated by cutting planes.

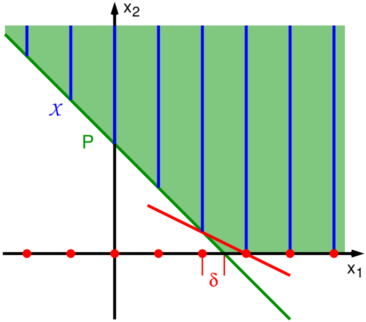
General Cutting Planes for MIP

Like in integer programming, “conv” is the best linear relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \le b, x_G \in \mathbb{Z}^G\}.$$

$Ax \le b$ is the LP-relaxation, P_G is approximated by cutting planes.

Ex.: Mixed Integer Rounding inequality (MIR)
simplest form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \ge \beta\}$, $\beta \in \mathbb{R}$



Put $\delta := \beta - \lfloor \beta \rfloor$,
then the inequ.

$$x_1 + \frac{1}{\delta} x_2 \ge \lceil \beta \rceil$$

is valid for $\mathcal{X}$.

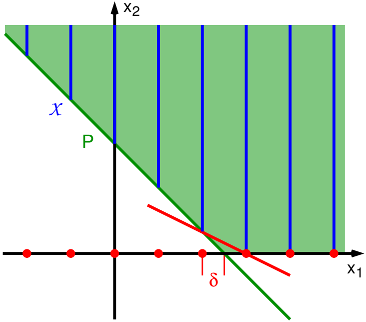
General Cutting Planes for MIP

Like in integer programming, "conv" is the best linear relaxation,

$$P_G := \mathrm{conv}\{x \in \mathbb{R}^n : Ax \le b, x_G \in \mathbb{Z}^G\}.$$

$Ax \le b$ is the LP-relaxation, P_G is approximated by cutting planes.

Ex.: Mixed Integer Rounding inequality (MIR)
simplest form: $\mathcal{X} = \{(x_1, x_2) \in \mathbb{Z} \times \mathbb{R}_+ : x_1 + x_2 \ge \beta\}$, $\beta \in \mathbb{R}$



Put $\delta := \beta - \lfloor \beta \rfloor$,
then the inequ.

$$x_1 + \frac{1}{\delta} x_2 \ge \lceil \beta \rceil$$

is valid for $\mathcal{X}$.

In state-of-the-art packages many further types are included (flow cover, cliques, etc.)

Branch-and-Cut Frameworks

When in a branch&bound-method the LP-relaxation of each subproblem is improved by cutting planes, this is called a Branch&Cut-Method. Currently these are the best methods for general mixed integer optimization.

Branch-and-Cut Frameworks

When in a branch&bound-method the LP-relaxation of each subproblem is improved by cutting planes, this is called a Branch&Cut-Method. Currently these are the best methods for general mixed integer optimization.

An efficient branch&cut implementation is subtle and difficult:
- choosing the next subproblem
- choosing the branching variable, fixing of variables
- storing subproblems efficiently in an incremental manner
- using cutting planes for several subproblems
- efficient heuristics for finding good feasible solutions

etc.

Branch-and-Cut Frameworks

When in a branch&bound-method the LP-relaxation of each subproblem is improved by cutting planes, this is called a Branch&Cut-Method. Currently these are the best methods for general mixed integer optimization.

An efficient branch&cut implementation is subtle and difficult:
- choosing the next subproblem
- choosing the branching variable, fixing of variables
- storing subproblems efficiently in an incremental manner
- using cutting planes for several subproblems
- efficient heuristics for finding good feasible solutions

etc.

There exist packages that provide the entire framework and allow to add further problem specific cutting planes and heuristics.

e.g. SCIP, Cplex, Gurobi, Abacus . . .